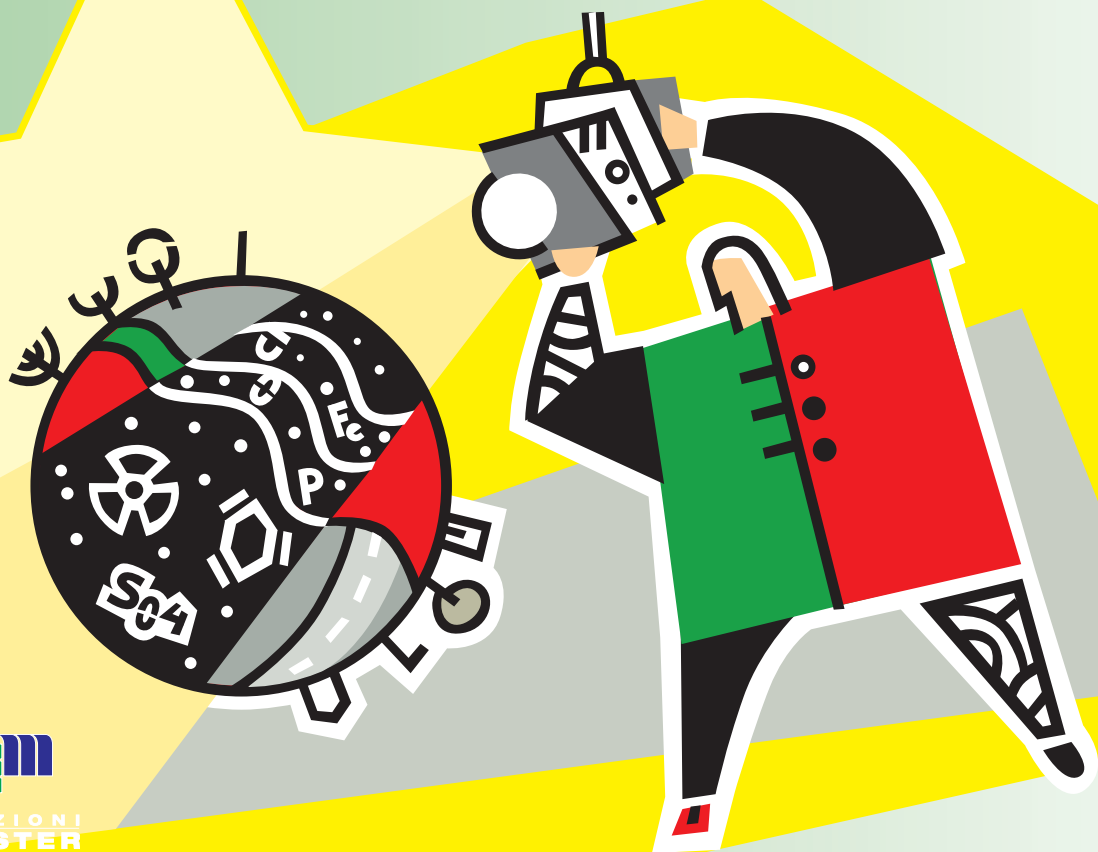
MASHUP: OVVERO COME INTEGRARE I SERVIZI OFFERTI DA GOOGLE, MICROSOFT, TECHNORATI, FLICKR E GLI ALTRI ALL'INTERNO DELLE PROPRIE APPLICAZIONI

LAVORARE CON INTERNET

Filippo Costalli & Ivan Venuti

EDIZIONI MASTER

LAVORARE CON INTERNET

di Filippo Costalli & Ivan Venuti

INDICE

PREFAZIONE

Questo libro presenta esempi e soluzioni che, grazie all'uso di diverse sorgenti di dati (ma tutte legate al Web), forniscono soluzioni nuove, integrate, personalizzate… in una parola: mashup (infatti è questo il termine, derivato dall'inglese, con cui si denota questo tipo di soluzioni).
Per farlo verranno analizzate diverse tecnologie, la cui scelta è dettata da più fattori. Hanno avuto una notevole rilevanza nella loro scelta anche la presenza di primitive e/o librerie per sfruttare meccanismi di comunicazione quali REST, Atom, RSS e la loro diffusione tra gli sviluppatori. Tutte queste considerazioni hanno portato a scegliere Java e PHP per lo sviluppo di esempi lato server. Questi linguaggi non verranno presentati, perché viene data per acquisita la loro conoscenza . Per quanto concerne la programmazione lato client è considerato un prerequisito l'uso di JavaScript. Ovviamente AJAX la fa da padrone per quanto concerne la comunicazione tra client e server. Anche in questo Google ha precorso i tempi fornendo una serie di librerie dall'eccezionale qualità, potenza e semplicità d'uso. Anche molti altri servizi, per la maggior parte gratuiti, vengono analizzati e usati negli esempi proposti. Chiaramente per ognuno non c'è spazio per approfondimenti e discussioni particolareggiate, ma per tutti sono fornite risorse e link che, insieme agli esempi, permettono di comprendere le funzionalità che si pensa possano aiutare a realizzare le proprie applicazioni.
Non resta che augurarvi una buona lettura con la speranza che il libro, oltre ad avervi aiutato a comprendere gli esempi presentati, possa aiutarvi a proporre vostri esempi originali ed applicazioni ancor più interessanti e innovative.

Filippo Costalli, Ivan Venuti
fcostalli@yahoo.it ,ivanvenuti@yahoo.it

Gli autori:

Filippo Costalli è laureato in Scienze dell'Informazione e lavora come analista programmatore. Appassionato di informatica da sempre, attualmente focalizza la sua attività sulla programmazione Object Oriented. È contattabile all'indirizzo fcostalli@yahoo.it. Ha scritto i Capitoli 2, 3, 5, 7.

Ivan Venuti è laureato in Informatica e lavora come analista programmatore. La passione per la programmazione lo ha sempre portato ad approfondire nuove tecnologie, tecniche di sviluppo e framework emergenti. Scrive abitualmente articoli per alcune tra le maggiori riviste di informatica italiane e ha al suo attivo numerosi libri sulla programmazione, in particolare su Java. Sul suo sito personale, raggiungibile all'indirizzo http://ivenuti.altervista.org, sono messe a disposizione notizie e informazioni aggiornate sulla sua attività professionale. Ha scritto i Capitoli 1, 4, 6, 8.

INTRODUZIONE

La prima ovvia domanda che ci si potrebbe porre è: "che si intende per mashup"? In questo caso ci facciamo aiutare da Wikipedia per una definizione un po' formale: "mash-up è letteralmente una poltiglia, in termini informatici è un'applicazione che usa contenuto da più sorgenti per creare un servizio completamente nuovo. Il contenuto dei mashup è normalmente preso da terzi via API, tramite feed (eg. RSS e Atom) o Javascript". Scomposta la frase, si analizzeranno alcuni termini che possono chiarire ancora meglio questa definizione e, in particolare:

- Applicazione
- Più sorgenti (di contenuto)
- API, Feed (RSS, Atom) o JavaScript

APPLICAZIONE

Un'applicazione implica la realizzazione di una soluzione completa, dotata sì di dati, ma anche (e soprattutto!) di funzionalità utili a raggiungere un determinato obiettivo. In questo contesto si dà per scontato che le applicazioni debbano essere usufruite via Web, quindi attraverso un opportuno browser. Web 2.0 è, aldilà di molte chiacchiere, un particolare modo di realizzare queste applicazioni. Da un lato c'è l'attenzione a realizzare applicazioni che rispondono in maniera simile alle usuali applicazioni desktop (quindi senza le interazioni submit/load tipiche delle pagine HTML del passato) ma dall'altro c'è una particolare attenzione all'utente e a renderlo partecipe delle informazioni reperite (o permettendo di aggiornarle o, perlomeno, di personalizzarne la visualizzazione).

PIÙ SORGENTI

Il concetto di sorgente dati è molto vago ma, per tutte, può essere con-

siderata un'origine remota, slegata dall'appliazione e da essa non controllata. In pratica chi realizza la sorgente dei dati non è (quasi mai) la stessa persona che realizza l'applicazione che ne estrae dati per eseguire il mashup. Per questo motivo è fondamentale definire e analizzare la modalità di estrazione e fruizione. Tra esse possono esserci:

- Accesso diretto alla pagina pubblicata (parsing HTML);
- Accesso attraverso feed (RSS o Atom)
- Accesso usando API (REST, SOAP, XML-RPC…) sul server
- Accesso usando API sul client (JavaScript)

Di seguito l'analisi delle diverse modalità.

Parsing HTML

La più semplice fonte di contenuto può essere una qualsiasi pagina Web. In questo caso è necessario far sì che la pagina venga analizzata alla ricerca delle informazioni cercate (probabilmente è necessaria un'operazione di parsing e di reperimento "mirato", ovvero specifico per la struttura della particolare pagina). È indubbio che questo è fonte di possibili problemi: nulla vieta che l'autore della pagina cambi completamente disposizione del contenuto, cambi l'ordine di visualizzazione o quant'altro. Ciò è perfettamente lecito in quanto una pagina Web è "pensata" per essere usufruita da una persona e non da un programma automatico. Ecco che chi ha interesse a condividere le informazioni, affinché vengano usate su siti di terze parti, ha la necessità di definire delle interfacce, che fungono da contratto per chi fornisce l'informazione e per chi la usa. Tali interfacce devono essere il più possibile standard e invarianti nel tempo, oltre che ad essere orientate ai dati (quindi al contenuto) e non alla loro presentazione (infatti, il fatto che un titolo sia in grassetto e di colore blu, il testo nero e font Verdana, è una scelta di presentazione; il fatto di identificare dei titoli e del testo, è una scelta di contenuto). Nel tempo sono state definite diverse forme di inter-

scambio, in particolare…

Feed RSS e Atom

Quasi tutte le tecnologie di interscambio dati sono basate su documenti XML. È il caso di feed RSS e Atom. RSS vede la nascita, come idea, nel 1995. Da allora si è evoluto fino a divenire uno standard di fatto per la distribuzione di aggiornamento, sia per contenuti pubblicati da portali che per blog o altri siti Web. Chi pubblica i dati mette a disposizione, in un documento XML standard, gli aggiornamenti delle informazioni in maniera da poter estrarre in modo incrementale le informazioni di interesse (grazie ad attributi come la data di pubblicazione o di aggiornamento). Per leggere gli RSS ci vuole un apposito software, chiamato lettore RSS o aggregatore RSS. Atom è un'evoluzione di RSS, nel senso che nasce come proposta di standard (si veda http://tools.ietf.org/html/rfc4287) e vuol risolvere il problema delle versioni RSS che nel tempo sono nate ed evolute (spesso in maniera poco controllata), ma mantiene inalterata l'idea di fondo di formato di interscambio di dati per la notifica degli aggiornamenti dei contenuti.

Feed RSS e Atom sono di enorme interesse sia per la quantità di siti Web che offrono questo tipo di informazione, sia perché Google usa questo formato (Google's Data API, in breve GData) per poter accedere agli aggiornamenti contenuti nei suoi servizi, quali Blogger, Google News, and Gmail e molti altri servizi che continua a sfornare in quantità (e qualità!) impressionante. GData, oltre ad essere un protocollo comune a tutti i servizi, è semplice da usare, in quanto Google mette a disposizione una libreria già pronta per i principali linguaggi di programmazione.

API da server (REST, SOAP, XML-RPC…)

Nel caso in cui le informazioni siano piuttosto complesse (dati eterogenei e a più dimensioni) e numerose (tanti dati) può essere conveniente definire delle modalità di estrazione più mirate rispetto a quan-

to sia possibile fare con i feed RSS (sia in termini di dati reperiti che in termini di interrogazioni tra le diverse dimensioni dei dati, quali la data, la presenza di sottostringhe in certi campi e così via). In questi casi si ricorre a tecnologie che vanno sotto il nome di Web Services. Anche in questo caso l'XML è la base per l'interscambio dei dati ma, nello specifico, è possibile definire anche operazioni esposte da un server e fruibili da uno o più client. Di solito si parla di applicazioni "business to business" per indicare che i fruitori dei Web Services (client) sono altre applicazioni e non utenti (come accade per un'interazione client/server classica com'è quella tra un browser e un server di pagine Web). In pratica i Web Service sono implementati da un'applicazione che risiede su un server che, per poter svolgere le sue attività, demanda alcune funzionalità (o reperisce ulteriori dati) ad altri server. Spesso si fa uso di frame work che semplificano la scrittura di Web Services (sia client che server). L'adozione di uno specifico frame work dipende sia dal linguaggio adottato che dalle caratteristiche volute. Nel seguito si farà uso di alcuni di questi frame work sia per Java che per PHP.

API da client (JavaScript)

Ultimamente emergono sempre più servizi che offrono le proprie funzionalità usando JavaScript, ovvero librerie che possono essere incluse dinamicamente dai browser che mostreranno le informazioni. Questa soluzione porta a estrarre i dati direttamente sul client. Si vedrà che questo può portare sia a vantaggi che svantaggi rispetto a soluzioni che fanno uso di Web Services lato server. Nel corso del libro verranno mostrati vari esempi che permettono di capire meglio questi aspetti.

SI DEVE PROGRAMMARE? NON È DETTO!

Una volta definito cosa si intende con mashup e chiarite quali pos-

sono essere le diverse tecnologie coinvolte, si può vedere, concretamente, come realizzare un semplice mashup. Per farlo non sono necessarie conoscenze di programmazione! Infatti, ultimamente, sono nati molti servizi (alcuni anche gratuiti!) per creare mashup usando semplici interfacce grafiche. In Tabella 1.1 alcuni dei principali siti che offrono questi servizi.

Sito Web	Descrizione
http://code.google.com/gme/	Google Mashup Editor: un servizio (ancora in fase beta) per la creazione di mashup
http://www.dapper.net	Permette di creare, e condividere, mashup a partire da siti Web esistenti. I dati possono essere esportati in vari formati (feed RSS, Google Maps, iCalendar e altro ancora).
http://www.bea.com/framework.jsp?CNT=index.jsp&FP=/content/products/aqualogic/ensemble/	BEA Aqualogin Ensemle: fare mashup usando tool di classe enterprise
http://www.popfly.com	Tool per mashup creato da Microsoft e che fa uso della nuova tecnologia silverlight

Tabella 1.1: Alcuni tra i più noti servizi che offrono la possibilità di creare mashup usando interfacce semplici e intuitive, senza ricorrere alla programmazione.

Per comprendere il tipo di risultati che si può ottenere si userà il servizio Dapper. Esso permette di estrarre dei contenuti da una o più pagine Web e di metterle a disposizione in vari formati (feed RSS, Google Maps, iCalendar e così via). Per prima cosa è opportuno identificare le pagine Web da cui estrarre i dati di interesse; successivamente bisogna capire come visualizzare i dati (pensando quali tra quelli a disposizione sono utili allo scopo) e si può iniziare con la creazione della "Dapp Factory" (è così che viene chiamata l'applicazione che si costruisce).

Un esempio con Dapper

La pagina http://www.inea.it/ssa/copindirizzi.html ha dei link a diverse pagine, tra cui una lista di università con la Facoltà di Agraria e quelle di Economia e commercio. La costruzione del nostro Dapp Factory consiste nello specificare la pagina di partenza e di scegliere il tipo di risultato; per esempio si potrà scegliere Google Maps, volendo mostrare una mappa con la georeferenziazione delle università reperite (Figura 1.1a).
Il passo successivo consiste nella preview della pagina (Figura 1.1b); da tale preview si può navigare tra i link e, usando il pulsante "Add to basket", si possono scegliere le pagine di interesse (Figura 1.1c). Il passo successivo permette di selezionare il contenuto da estrarre (Figura 1.1d).

Figura 1.1: Dapper e la creazione di un Dapp Factory.

L'estrazione del contenuto, nel caso di tabelle, è piuttosto semplice: basta esitare la tabella con l'apposito pulsante ("Table") e fare clic sulla lettera della colonna (che, come in un foglio Excel, è "A", "B"

e così via). Il primo dato che si vuol estrarre è il nome dell'università (colonna A). Una volta selezionata la colonna si avrà sulla finestra "Preview selected content" la lista delle università della pagina. Per rendere persistente la selezione premere sul pulsante "Save fields" dandogli un nome significativo (per esempio "Ente"). Per l'indirizzo è necessario selezionare più colonne, in particolare le colonne B (indirizzo), C (Cap) e D (Città). Per la selzione multipla basterà tenere premuto il tasto Ctrl. Anche per queste informazioni ci sono la preview e la possibilità di assegnarle ad un campo (in questo caso lo si può chiamare "indirizzo"), come mostrato in Figura 1.2.

Figura 1.2: Estrazione degli indirizzi.

I due campi (ente ed indirizzo) in realtà si riferiscono alla stessa informazione; per questo nella pagina successiva li si può selezionare e raggruppare. Il passo seguente permette di salvare le informazioni (per farlo è necessario fornire un indirizzo di email valido e proteggere l'account con una password). Poi si può scegliere il nome dell'applicazione da salvare, associargli una descrizione e decidere se

lasciarla pubblica o renderla privata (Figura 1.3) C'è anche la possibilità di segnalare che quella creata è un'applicazione di test: in questo caso dopo 24 ore essa verrà eliminata dal sistema.

Figura 1.3: Caratteristiche di pubblicazione.

Terminata la creazione, la nuova applicazione è sempre accessibile sia dal proprio profilo (da cui la si può modificare) o come link diretto. In ogni caso la mappa presenta tanti marcatori quanti sono le università reperite; ciascuno è posizionato in corrispondenza dell'indirizzo indicato e, facendoci clic, appare un fumetto con la descrizione reperita dalla pagina Web di origine (Figura 1.4).

Figura 1.4: La mappa generata usando gli indirizzi reperiti sul Web.

NON BASTA?

Benché i servizi di mashup per non programmatori stiano divenendo sempre più complessi e completi, il loro utilizzo non giustifica certo la scrittura di un intero libro. D’ora in avanti si farà uso di programmi, sia client che server, mostrando i passi per creare applicazioni che realizzano mashup. Il vantaggio di ricorrere a questo tipo di soluzioni è la flessibilità: si mostreranno casi in cui si possono usare alcune tecnologie, altre volte le si integrerà con altre e più sofisticate librerie ma, in tutti i casi, le soluzioni possono essere modificate e adattate alle proprie esigenze. In Rete, come sempre, esistono numerosi servizi che forniscono API. Un buon punto di inizio può essere ProgrammableWeb (, Figura 1.5). Però, prima di programmare, è necessario aver configurato opportunamente il sistema e installato gli strumenti necessari…

Figura 1.5: Il sito ProgrammableWeb, dove sono censiti centinaia di API e mashup esistenti.

L’AMBIENTE DI SVILUPPO

In questo capitolo verranno intodotte le tecnologie utilizzate nel prosieguo del libro. Particolare attenzione sarà posta nell’illustrare come creare un proprio ambiente di sviluppo per utilizzare, estendere e migliorare le applicazioni che, in seguito, verranno presentate e descritte.

Attenzione
Per poter eseguire gli esempi è necessario avere installato (e funzionante) un Apache Tomcat 5 e un Web Server che interpreti PHP 5. Se si hanno già disposizione questi strumenti potete saltare questo capitolo e installare da subito le applicazioni. Inoltre, alla pagina http://ivenuti.altervista.org/risorse/mashup.htm, sono presentati tutti i link del libro, al fine di facilitarne l’uso senza doverli riscrivere uno ad uno.

JAVA E LE APPLICAZIONI WEB

Sul proprio computer è necessario far sì che venga installato un ambiente adatto all’installazione di applicazioni Web realizzate con Java (applicazioni che in gergo sono chiamate “webapp”, contrazione di “webapplication”). Per farlo si consiglia di scaricare il JDK della Sun (che offre gli strumenti per compilare ed eseguire le classi Java) e Apache Tomcat (che è il server di riferimento per le applicazioni Web Java). Di seguito le istruzioni minime per la loro installazione. Si rimanda ai siti ufficiali per eventuali approfondimenti sul loro uso e la loro configurazione avanzata.

Installare il JDK

Collegandosi alla pagina http://java.sun.com/j2se/index.jsp è possibile seguire il link verso il Java Development Kit (abbreviato in JDK). È possibile scaricare diverse versioni; per gli esempi di questo libro è necessario usare una versione 5.0.12 (la versione di “Update”, ovvero l’ulti-

mo numero, è opzionale; si consiglia di scaricare sempre l'ultima, che al momento di andare in stampa è, appunto, la 12). Il JDK è disponibile come installer specifico per i sistemi operativi (Windows e Linux).
Settare le variabili d'ambiente
Una volta installato il JDK è opportuno settare opportunamente alcune variabili d'ambiente; benché questa operazione non sia sempre fondamentale, è vivamente consigliata, sia perché in questo modo è più comodo utilizzare gli strumenti del JDK, sia perché molte applicazioni, che si basano su Java, fanno uso di alcune di queste informazioni (è il caso di Apache Tomcat). Per conoscere le variabili d'ambiente già presenti basta aprire una console di comandi e, in Windows, digitare:

```
set
```

in Linux, invece, si digiterà il comando:

```
env
```

In entrambi i casi vengono stampate a video tutte le variabili d'ambiente già definite (in Figura 2.1 un esempio).

Figura 2.1: risultato del comando set su un sistema Windows.

Per agevolare lo sviluppo di programmi Java si definiscano le variabili d'ambiente JAVA_HOME e CLASS_PATH in maniera che la prima indichi la cartella di installazione del JDK, la seconda dove trovare le classi (inizialmente i due valori coincideranno). È bene provvedere anche a modificare la variabile PATH affinché vengano aggiunti gli eseguibili del JDK. In Tabella 2.1 le variabili da definire per i sistemi Windows e in Tabella 2.2 quelle per Linux, con la descrizione del loro significato e i valori consigliati.

Variabile	Valore (consigliato)	Significato
JAVA_HOME	set JAVA_HOME=c:\cartella\path\	Cartella di installazione del JDK
CLASSPATH	set CLASSPATH=.;%JAVA_HOME%	Sia la cartella di installazione del JDK che quella corrente (indicata con il carattere "punto": ".") e che è consigliabile mettere per prima)
PATH	set PATH=%PATH%;%CLASSPATH %\bin	Mantenere il PATH definito di default dal sistema e aggiungervi anche quello della cartella /bin del JDK

Tabella 2.1: Variabili d'ambiente da settare (Windows); si noti l'uso del carattere ";" per separare una lista di valori e di %nome% per riferirsi ad una variabile definita in precedenza.

Le variabili sono definibili direttamente da una console di comandi ma così facendo hanno un valore locale ad essa (in pratica vengono perse quando si chiude la console corrente e, similmente, aprendo nuove console le variabili definite nella prima non sono propagate alle altre). Ovviamente è comodo, a parte casi particolari, definirle una volta per tutte.

In Windows Xp le variabili d'ambiente si definiscono, in maniera permanente, ovvero che persiste anche nelle successive sessioni, facendo clic

Variabile	Valore (consigliato)	Significato
JAVA_HOME	set JAVA_HOME=c:\cartella\path\	Cartella di installazione del JDK
CLASSPATH	set CLASSPATH=.;%JAVA_HOME%	Sia la cartella di installazione del JDK che quella corrente (indicata con il carattere "punto": ".").
PATH	set PATH=%PATH%;%CLASSPATH %\bin	Mantenere il PATH definito di default dal sistema e aggiungervi anche quello della cartella /bin del JDK

Tabella 2.2: Variabili d'ambiente da settare (Linux); si noti l'uso del carattere ":" per separare una lista di valori e di $nome per riferirsi ad una variabile definita in precedenza.

con il pulsante destro su "Risorse del sistema", scegliendo "Proprietà" dal menu contestuale. Dalla finestra si selezioni il tab "Avanzate" e quindi il pulsante "Variabili d'ambiente".
In Linux invece è necessario editare il file di configurazione della shell in uso. Se essa, com'è di solito, è la bash, bisognerà editare il file .bash-profile presente nella cartella home dell'utente con cui ci si collega (tale cartella è quella a cui si accede eseguendo il comando "cd" senza parametri).

Attenzione
Se si vuol visualizzare il valore di una variabile d'ambiente è necessario aprire una console di comandi e digitare il comando echo %nomevariabile% in Windows, mentre il comando echo $nomevariabile è utilizzabile sui sistemi Linux.

Si faccia attenzione anche al fatto che in Linux tutti i comandi e i nomi di variabili di sistema sono "case sensitive": è importante rispettare le

maiuscole/minuscole dei nomi; in Windows, invece, maiuscolo o minuscolo è lo stesso (in questo caso si dice che le variabili, e i loro valori, sono "case insentitive").

Test dell'ambiente

Una volta definite le variabili sopraccitate, si apra una console dei comandi e si provi a digitare:

```
java –version
```

se tutto è andato a buon fine dovrebbe apparire la versione dell'interprete Java (versione che coincide con quella del JDK installato, Figura 2.2).

```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\IdT>java -version
java version "1.5.0_06"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_06-b05)
Java HotSpot(TM) Client VM (build 1.5.0_06-b05, mixed mode, sharing)

C:\Documents and Settings\IdT>_
```

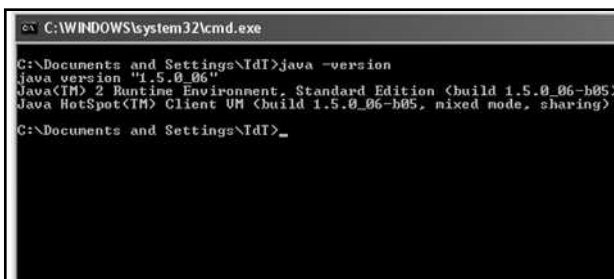



Figura 2.2: Verifica che il JDK sia configurato correttamente.

Installare Tomcat

La pagina di riferimento del progetto è *http://jakarta.apache.org*. Come si può notare esistono diverse versioni di Tomcat; ciascuna implementa una specifica versione delle specifiche Servlet e JSP. Per eseguire gli esempi del libro è possibile usare sia la versione 6 (anche se, al momento di andare in stampa, è ancora considerata un release non stabile), sia la versione 5.5.
Tutti i pacchetti iniziano con jakarta-tomcat-[versione], dove [versione] è il numero di versione. Poi ci possono essere un ulteriore suffisso e una

diversa estensione, in particolare:

.zip: distribuzione di base (in formato compresso);
.exe: contiene il pacchetto e l'installer per Windows;

Tutte le versioni (ovviamente a meno della .exe) sono anche in formato .tar.gz per essere utilizzate con il comando tar (compatibile con la versione GNU).

Installazione di Tomcat

Durante l'installazione (completamente guidata nel caso si scelga l'installer di Windows) è necessario prestare attenzione alle scelte:

1. quali componenti includere
2. il path del JRE
3. porta ove risponde il server e credenziali di amministratore

Per i componenti si consiglia la versione "Full", comprensiva di tutti i componenti. Il path del JRE deve contenere il percorso per accedere al JRE (precedentemente installato e deve avere una versione 5.0 o successiva), mentre la port è, di default, la 8080, anche se se ne può scegliere una diversa; tale scelta modificherà la url di connessione alle applicazioni installate. Inoltre si possono impostare le credenziali dell'amministratore (esso è l'utente che può gestire in toto la configurazione e la manutenzione del server).

Verificare che tutto funzioni

La verifica dell'avvenuta installazione è molto semplice: basta collegarsi all'url del sistema locale (127.0.0.1 o utilizzando l'alias "localhost") e specificare la porta impostata nella fase di installazione (di default la 8080); se tutto funziona correttamente apparirà una pagina predefinita come quella mostrata in Figura 2.3 (i dettagli possono differire per versioni di Tomcat diverse).

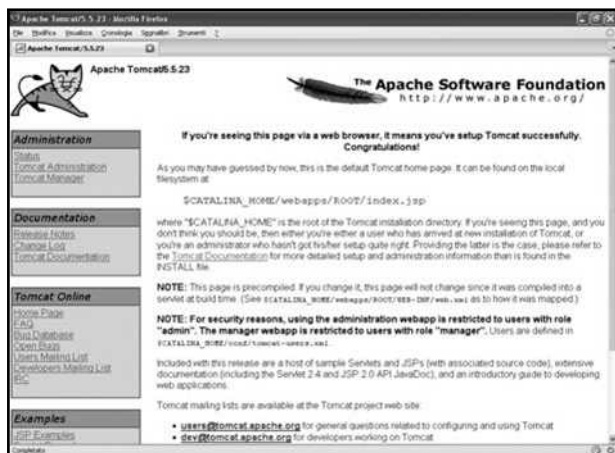



Figura 2.3: la pagina di benvenuto dopo aver installato Tomcat correttamente.

Le cartelle di Tomcat

In Figura 2.4 le diverse cartelle contenute nella root dell’installazione (d’ora in poi si farà riferimento alla root con il nome $CATALINA_HOME: è la stessa convenzione utilizzata in tutta la documentazione ufficiale a corredo di Tomcat).

Figura 2.4: le cartelle (a partire dalla root dell’installazione di Tomcat).

Attenzione
Una volta installato il Tomcat si possono provare gli esempi del libro copiando il file con estensione .war direttamente nella cartella webapps/.

Le più importanti sono webapps/, che conterrà tutte le applicazioni, logs/ che conterrà i log (stampe) sia di Tomcat che delle applicazioni e bin/, che contiene gli script per l'esecuzione e lo stop del Tomcat.

JAVA E I WEB SERVICES

I Web Services sono particolarmente semplici da usare in quanto chi crea il servizio espone un documento, chiamato WSDL, che descrive tutti i dettagli del servizio. Grazie ad esso è possibile creare l'infrastruttura di comunicazione in maniera automatizzata. Per farlo si deve però utilizzare un frame work apposito o delle primitive del linguaggio usato.
Java è particolarmente ricco di strumenti di sviluppo per realizzare Web Services. Basti pensare che l'ultima release del linguaggiom, la 6.0, offre supporto nativo alla loro realizzazione. Per quanto concerne tool di terze parti non c'è che l'imbarazzo della scelta. Tra i frame work "storici" più utilizzati si segnala senz'altro Axis (http://ws.apache.org/axis/) ma non mancano altre proposte, sia Open Source che commerciali. In Tabella una sintesi dei principali frame work.

Sito Web	Framework
http://java.sun.com/javase/6/	La distribuzione 6.0 ha il supporto dei Web Services inclusa direttamente nel JDK
http://ws.apache.org/axis/	Axis: uno dei più diffusi framework Open Source per Java (esiste un porting anche per C++)
http://ws.apache.org/axis2/	Axis 2: la versione successive di Axis, che è una completa riscrittura e re-ingegnerizzazione

Sito Web	Framework
http://xfire.codehaus.org/	XFire: un frame work particolarmente performante che sta riscuotendo notevole successo (anch'esso Open Source)
http://www.systinet.com/products/ssj/overview	Systinet Server for Java; un server commerciale per WS in Java
http://xins.sourceforge.net/	XINS (XML Interface for Network Services); progetto Open Source che supporta SOAP, XML-RPC e REST
http://labs.jboss.com/jbossws/	JBossWS: un sottoprogetto di JBoss specifico per I Web Services
http://www.restlet.org	Un framework per applicazioni che usano REST (sia client che server)

Tabella 2.3: alcuni framework per sviluppare Web Services in Java.

Attenzione
I Web Services sono un argomento particolarmente sensibile alle evoluzioni, sia tecnologiche che per quanto concerne gli standard di riferimento. Non da meno è importante usare sempre le nuove versione dei diversi framework. Java 6, per esempio, ha rilasciato un aggiornamento poco dopo il rilascio della versione ufficiale (https://jax-ws.dev.java.net/). XFire sta evolvendo in un progetto Apache e attualmente è in fase "incubator", ovvero in attesa di entrare a far parte dei progetti ufficiali. Il nome del nuovo progetto sarà Apache CXF (sito di riferimento http://incubator.apache.org/cxf/)

L'applicazione Java allegata

Il libro descriverà varie applicazioni realizzate in Java (e, come si vedrà successivamente, anche PHP). Per l'installazione delle applicazioni Java è sufficiente copiare il file LibroMashup.war sotto la cartella webapps/ di Tomcat. Accedendo all'indirizzo http://server:porta/LibroMashup apparirà la pagina iniziale (Figura 2.5) contenente i link a tutte le applicazioni.

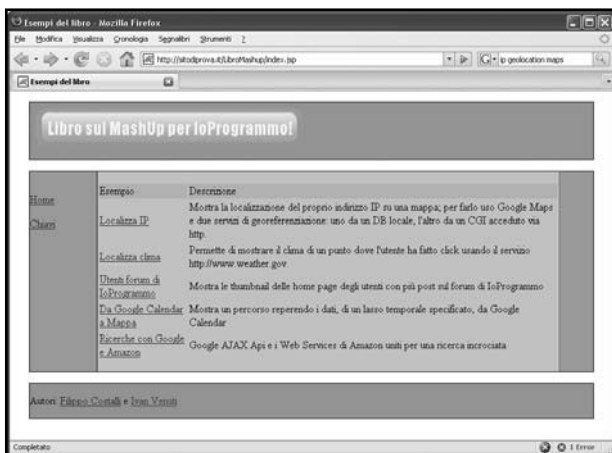



Figura 2.5: la pagina principale del progetto Java con gli esempi del libro.

Prima di poterle usare è necessario ottenere varie chiavi di accesso (una per ogni servizio). Tali chiavi sono configurabili seguendo il link "Chiavi" (Figura 2.6).

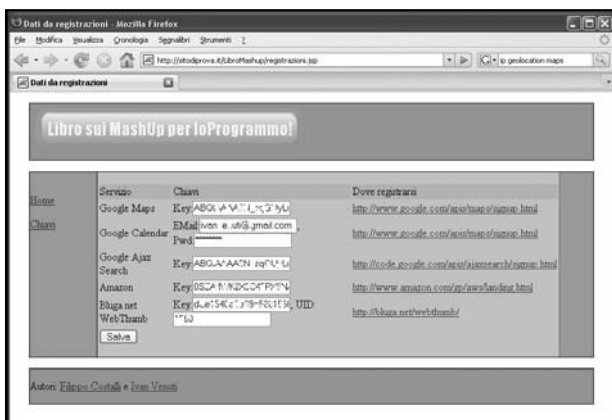



Figura 2.6: la pagina dove inserire le chiavi dei diversi servizi.

EASYPHP

Per l'utilizzo del linguaggio PHP adotteremo il prodotto EasyPHP, contenente tutto il necessario per un utente che desideri iniziare immediatamente a sviluppare con PHP. EasyPHP è un pacchetto software che consente l'installazione del server HTTP Apache, del database MySQL ed il completo supporto del linguaggio di scripting PHP. Il tutto integrato in un unico prodotto. Vediamo come procurarsi questo prodotto, come installarlo sulla nostra macchina e come utilizzarlo.

Installazione

Per prima cosa occorre scaricare il pacchetto, collegandosi all'indirizzo http://www.easyphp.org/telechargements.php3 La procedura di installazione risulta essere notevolmente semplice ed intuitiva. È sufficiente, infatti, cliccare sull'icona del programma e premere "avanti" ogni qual volta questo venga chiesto. Una volta terminata questa semplice operazione, verificare la corretta installazione nel seguente modo.
Avviare il pacchetto cliccando sul collegamento presente nel menu "Start/Programmi/EasyPHP"; effettuata questa operazione dovrebbe apparire un'icona nella barra di sistema vicino all'orologio. Fare clic su questa per accedere al menù e selezionare l'opzione 'Avvia'.
Apparirà a video una finestra simile a quella che segue, che segnala il corretto avvio del server Apache e del db MySQL (Figura 2.7).

Figura 2.7: Avvio di EasyPHP

Aprire una finestra di browser e collegarsi all'indirizzo http://127.0.0.1/ (l'indirizzo IP con il quale la macchina chiama se stessa). Nel caso in

cui tutto sia stato eseguito correttamente, dovrebbe comparire la scritta di presentazione del prodotto.

Esecuzione di script con EasyPHP.

Una volta scritto un programma PHP, bisogna fare in modo che questo possa essere eseguito da EasyPHP. Occorre, per questo, posizionare i files sotto la cartella "www" presente sotto PHP. Creare, sotto la directory "www", una cartella per ogni progetto che desideriamo eseguire, e sotto ognuna di queste, posizionare i files appropriati.
Lanciare PHP e connettersi, tramite browser, all'indirizzo http://127.0.0.1 o http://localhost , EasyPHP proporrà automaticamente una pagina di partenza con la lista dei progetti che abbiamo disposto sotto "www".

Easy PHP e PHP5

Negli esempi che andremo ad illustrare nel corso del libro verrà utilizzata la versione 5 di PHP. La scelta è stata fatta perché PHP5 migliora notevolmente i punti deboli delle versioni precedenti del linguaggio. Esso offre un completo supporto per la programmazione ad oggetti avvicinandosi, così, ai concetti espressi da Java. Inoltre – ed è questa la caratteristica che, come vedremo, verrà sfruttata maggiormente - è stata introdotta la possibilità di estrarre informazioni da documenti XML, di cui si conosca la struttura, in maniera immediata e con l'utilizzo di pochissime righe di codice. Questa nuova estensione si chiama SimpleXML e viene trattata nel paragrafo successivo.
EasyPHP supporta PHP5 a partire dalla versione 2.0 che, nel momento in cui scriviamo, è ancora in versione beta. È comunque possibile includere il supporto PHP5 anche nella versione 1.8, l'ultima distribuzione stabile.
Per fare ciò è sufficiente scaricare la versione 5 di PHP dal sito http://www.php.net/downloads.php seguendo il cammino Windows binaries > PHP 5.2.3 zip package ed operare come segue:
1. Creare una cartella php5 sotto la root di EasyPHP (supponiamo che sia c:\EasyPHP\)

2. Decomprimere sotto tale cartella lo zip scaricato

3. Modificare il file httpd.conf, situato sotto la directory c:\EasyPHP\apache\conf\ ,come segue:

   a. Rimpiazzare LoadModule php4_module "c:/EasyPHP/php/php4apache.dll" con LoadModule php5_ module "c:/EasyPHP/php5/php5apache.dll"

   b. Rimpiazzare AddModule mod_php4.c con AddModule mod_php5.c

   c. Aggiungere php5 alla riga DirectoryIndex index.html index.shtml index.wml index.pwml index.php index.php3 index.php4 index.php5

   d. Aggiungere .php5 alla riga AddType application/x-httpd-php .phtml .pwml .php3 .php4 .php5 .php .php2 .inc

4. Modificare il file C:/EasyPHP/php.ini come segue:

   a. Rimpiazzare extension_dir = "${path}\php\extensions\" con extension_dir = "${path}\php5\ext\"

   b. Rimpiazzare ;include_path = "${path}\php\pear\" con include_path = "${path}\php5\PEAR\"

   c. Aggiungere la riga extension=php_mysql.dll

5. Copiare il file c:\EasyPHP\php5\libmysql.dll nella directory c:\windows\system32\

6. Avviare EasyPHP

A questo punto anche la nostra installazione EasyPHP supporta PHP5. Non resta, adesso, che passare ad introdurre l'utilizzo di SimpleXML, l'estensione per la gestione XML cui avevamo fatto cenno in precedenza e che è abilitata di default all'interno del motore PHP5.

SIMPLEXML, GESTIONE XML CON PHP5

Con il termine parsing di un documento XML intendiamo tutte le azioni di accesso, modifica e interrogazione del documento. La libreria SimpleXML (in Figura 2.8 il sito di riferimento) possiede una interfaccia ad oggetti semplice ed intuitiva, che permette, con pochissime righe di codice, di accedere agli elementi interessati seguendo la struttura del file XML anche nella rappresentazione interna.
Vediamo, a mò di esempio, come parsare un documento XML con SimpleXML.

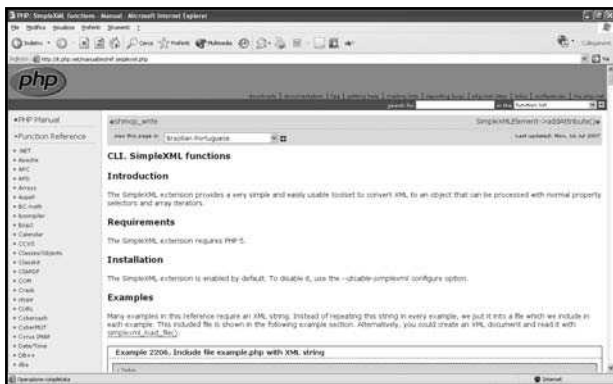



Figura 2.8: Il sito di riferimento della libreria SimpleXML.

```
<?xml version='1.0' standalone='yes'?>
<biblioteca>
```

```
<libro>
  <titolo>I promessi Sposi</titolo>
  <autore>Alessandro manzoni</autore>
  <argomento>Romanzo Storico</argomento>
  <informazioni>
    <editore>Mondadori</autore>
    <prezzo>10</argomento>
  </informazioni>
</libro>
<libro>
  <titolo>La ragazza di Bube</titolo>
  <autore>Carlo Cassola</autore>
  <argomento>Romanzo</argomento>
  <informazioni>
    <editore>Rizzoli</autore>
    <prezzo>8</argomento>
  </informazioni>
</libro>
<libro>
  <titolo>Apocalittici e Integrati</titolo>
  <autore>Umberto Eco</autore>
  <argomento>Saggio</argomento>
  <informazioni>
    <editore>Bompiani</autore>
    <prezzo>8</argomento>
  </informazioni>
</libro>

</biblioteca>
```

SimpleXML utilizza la funzione simplexml_load_string() per l'accesso ad un documento XML. Viene caricato, all'interno di un oggetto SimpleXML, il contenuto del file XML passato come parametro. L'oggetto restituito

rappresenta la root del documento XML ed espone un attributo per ogni tag figlio, attributo che può essere un array (nel caso ci siano più tags omonimi come figli) o singolo elemento. L'accesso a tali attributi viene effettuato utilizzando la sintassi consuete per l'accesso agli elementi di un array.
Per esempio, supponendo che il documento XML sia caricato nella variabile $biblioteca_xml, per visualizzare a video il primo autore presente nell'elenco è sufficiente il codice:

```
$biblio = simplexml_load_string($biblioteca_xml);
echo $biblio ->libro[0]->autore;
```

Tramite a funzione simplexml_load_string() è possibile anche effettuare il parsing di entità multiple all'interno della gerarchia XML. Per esempio, per isolare la lista completa degli autori, è sufficiente ricorrere ad un semplice ciclo iterativo:

```
$biblio = simplexml_load_string($biblioteca_xml);
foreach ($biblio->libro as $libro)
{
  echo $libro->autore.'<br />';
}
```

Abbandoniamo, per ora, SimpleXML e passiamo a presentare l'ultimo strumento di lavoro, che ci permetterà di effettuare chiamate a servizio remoti: LibCurl.

LIBCURL, UNA LIBRERIA PER COLLEGAMENTI REMOTI

LibCurl è una libreria, creata da Daniel Stenberg, che permette la comunicazione con macchine remote utilizzando una vasta gamma di protocolli di rete (in Figura 2.9 l'home page del progetto). Al momento ven-

gono supportati i seguenti protocolli: http, https, ftp, gopher, telnet, dict, file, e ldap. Oltre a questo vengono gestiti anche certificati HTTPS ed è possibile utilizzare il protocollo ftp per l'upload di files.
EasyPHP contiene l'estensione per libCurl; per abilitarla è sufficiente aprire il file php.ini ed andare a decommentare la riga ;extension=php_curl.dll.

Figura 2.9: Home page del progetto LibCurl: *http://curl.haxx.se/.*

Per iniziare una sessione di curl utilizzare l'istruzione curl_init(). Successivamente settare le opzioni per la sessione tramite la funzione PHP curl_setopt(). Una volta settate le proprietà desiderate eseguire la richiesta con currl_exec(), e chiudere la sessione.

```
<?
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'http://www.example.com');
curl_setopt($ch, CURLOPT_HEADER, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
$data = curl_exec();
curl_close($ch);
?>
```

Nell'esempio sono state settate tre proprietà: CURLOPT_URL corrisponde all'url a cui collegarsi; CURLOPT_HEADER per specificare se deve essere ritornato l'header della risposta del server; CURLOPT_RETURNTRANSFER settato ad 1 in modo che una risposta venga ritornata. Questa viene memorizzata nella variabile $data.

API DI GOOGLE

Sul Web è possibile trovare moltissime risorse che forniscono dati utili a creare dei mashup. Tra le risorse di maggior interesse c'è il già citato ProgrammableWeb che è specifico per i mashup. Per quanto concerne le collezioni di Web Services, si può trovare in http://www.strikeiron.com e http://www.xmethods.net un'ottima collezione di servizi. Però Google ha una serie di servizi di prim'ordine; il più interessante è senz'altro Google Maps…

GOOGLE MAPS

Google Maps ha rivoluzionato, sin dalla sua introduzione, il modo con cui gli utenti consultano mappe on-line. Ora si analizzerà la struttura di questa applicazione fornendo anche un primo esempio della semplicità con cui è possibile integrarla nel proprio sito web; successivamente le Google Maps saranno usate in moltissimi esempi del libro.

Che cosa sono

Per Google Maps si intende un servizio, rilasciato da Google, che permette agli utenti di cercare e visualizzare, all'interno di un browser, mappe geografiche che coprono quasi tutta la terra. Il servizio è accessibile via web al sito *http://maps.google.it/* ed è strutturato tramite mappe dinamiche di semplice utilizzo. Le cartine fornite possiedono interessanti caratteristiche interattive; per esempio basta tenere premuto il mouse su un qualsiasi punto per spostarsi sulla mappa a proprio piacimento ed inoltre, tramite un intuitivo sistema di bottoni a forma di frecce, si riesce, in maniera molto semplice, ad aumentare o diminuire il dettaglio di visualizzazione, effettuando veri e propri zoom.
Un'altra caratteristica estremamente interessante di Google Maps è quella di rendere disponibile, oltre alla mappa stradale, anche la mappa satellitare (ottenuta utilizzando il motore Google Hearts). Le due tipologie di mappa sono visualizzabili separatamente, ma è possibile an-

Figura 3.1: esempio di Google Maps

che sovrapporle, visualizzando, in questo modo, sia la toponomastica delle strade che la loro reale morfologia.
Oltre a questo è possibile trovare il percorso migliore per andare da un luogo ad un altro (utilizzando così il servizio come un vero e proprio navigatore satellitare) oppure ricercare e localizzare, sulla mappa, servizi quali ristoranti, negozi, luoghi turistici o istituzionali.
Recentemente Google ha aggiunto una nuova prospettiva alle proprie mappe: è disponibile, infatti, il servizio denominato "Street View", che permette di esplorare le città non più dall'alto ma direttamente dal livello stradale, dando l'impressione all'utente di stare camminando per la città. Insomma, vere e proprie camminate virtuali, con la possibilità di effettuare zoom e di scegliere il percorso decidendo, di volta in volta, se girare a destra o a sinistra.
Dopo aver brevemente illustrato in cosa consistono le Google maps, iniziamo ad esplorarne la struttura tramite un primo, semplice esempio di integrazione di questo servizio in una nostra pagina web.

Google Maps API: come funziona?

Per l'inserimento e la gestione delle proprie mappe all'interno di siti web, Google mette a disposizione degli sviluppatori un insieme di procedure scritte in Javascript, dette Google Maps API, la cui documenta-

zione dettagliata è consultabile all'indirizzo *http://www.google.com/apis/maps/documentation/*.
Si tratta di uno strumento gratuito che offre una serie di funzioni per la manipolazione e gestione delle mappe e che, soprattutto, permette di inserire, in maniera agevole, Google Maps nelle proprie pagine Web tramite Javascript. Queste API sono di facile utilizzo ed estremamente versatili e potenti; nei paragrafi successivi vedremo come sia possibile utilizzarle per creare una mappa impiegando solo poche righe di codice JavaScript.
Per poter utilizzare le API Google Maps all'interno del proprio sito web è però necessario ottenere una Google Maps API Key, che va richiesta direttamente a Google. Vediamo di che si tratta e come fare per procurarsela.

Ottenere una chiave

Il servizio Google Maps è gratuito ma occorre procurarsi una chiave per l'utilizzo delle API, quella che, comunemente, viene chiamata API Key.
Una API Key altro non è che un codice alfanumerico che identifica univocamente l'URL (o una sottopagina di questo URL) che andrà ad utilizzare le mappe.
Per ottenere una API Key è sufficiente collegarsi alla pagina http://maps.goo-

Figura 3.2: Form per la richiesta di Api Key

gle.com/apis/maps/signup.html, indicare l'indirizzo per il quale si ha intenzione di utilizzare il servizio (per es. http://www.miosito.it/) e premere il bottone Generate API Key.
Verrà generata una stringa di lettere e numeri piuttosto lunga simile a quella riportata nella figura sottostante. Salvatela in un file di testo poiché ci servirà a breve.

Thank you for signing up for a Google Maps API key!

Your key is:

```
ABQIAAAA0V8JRNk2StB-h9e4f0ucsRSz5UsnpVeLRWcYtI0oPzakgHbG3BT6VbOrOyT8VI9CytI1EoGbMIt0SQ
```

This key is good for all URLs in this directory:

```
http://www.miosito.it/
```

Here is an example web page to get you started on your way to mapping glory:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps JavaScript API Example</title>
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAA0V8JRNk2StB-
h9e4f0ucsRSz5UsnpVeLRWcYtI0oPzakgHbG3BT6VbOrOyT8VI9CytI1EoGbMIt0SQ"
      type="text/javascript"></script>
    <script type="text/javascript">
    //<![CDATA[
    function load() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map"));
        map.setCenter(new GLatLng(37.4419, -122.1419), 13);
      }
    }
    //]]>
    </script>
  </head>
  <body onload="load()" onunload="GUnload()">
    <div id="map" style="width: 500px; height: 300px"></div>
  </body>
</html>
```

Figura 3.3: la chiave generata da Google

È importante sottolineare che la chiave così ottenuta è valida solo per il dominio richiesto, per cui se sviluppate la vostra applicazione su di un server locale (su dominio localhost) e, successivamente, vorrete effettuare il trasferimento dei files su di un server di produzione, dovrete procurarvi due differenti API keys. Detto questo, vediamo come utilizzare la chiave ottenuta per creare una mappa e comprendere i concetti fondamentali della Google Maps API.

Impostazione della prima mappa

Come avrete notato, insieme alla chiave viene fornito anche un primo, semplice esempio di come dovrebbe essere strutturata una pagina web

contenente una Google Maps. Copiamo l'esempio (viene fornito per questo, ovviamente) sostituendo la chiave con quella fornitaci da Google e salviamo il codice sul file esempio1.html.
A questo punto diamo un'occhiata al codice fornitoci da Google in modo da comprendere i meccanismi di base dell'utilizzo delle API. Tutta la logica sta dentro la funzione Javascript load(), che consiste nelle due righe di codice Javascript riportate nella porzione di codice sottostante.

```
var map = new GMap2(document.getElementById("map"));
map.setCenter(new GLatLng(37.4419, -122.1419), 13);
```

Proviamo a capirne il significato.
La prima riga instanzia un oggetto Javascript di tipo GMap2 e lo assegna alla variabile map. L'oggetto GMap2 rappresenta la mappa vera e propria. Quindi in questo modo si crea la mappa e la si pone, nella pagina HTML, all'interno della sezione contrassegnata dal tag "DIV" che ha come identificativo il nome "map".
Nella seconda riga viene utilizzato metodo setCenter() che si occupa di "puntare" la mappa su una determinata locazione geografica. Vediamo come questa operazione viene effettuata.
Il metodo setCenter accetta due parametri: un punto geografico ed il livello di zoom con cui la mappa sarà visualizzata. Nelle api Google Maps il concetto di punto è espresso dalla funzione GLatLng() che accetta a sua volta due parametri: la latitudine e la longitudine in formato decimale. Per cui la funzione che stiamo esaminando centra la mappa sulle coordinate geografiche 37.4419 e –122.1419 con un livello di zoom pari a 13.
Potete provare l'esempio sostituendo le coordinate fornite da Google con quelle delle località che preferite, le cui coordinate possono essere reperite utilizzando il servizio fornito da http://world.maporama.com
Nel corso del libro, comunque, impareremo ad ottenere in maniera dinamica le coordinate geografiche a partire da una località, da un indirizzo o dall'IP di un server.

Tornando al nostro esempio, nel corso dell'introduzione si era accennato alla possibilità di muoversi ed effettuare zoom sulla mappa nonché di visualizzare, in alternativa alla mappa stradale, la mappa satellitare oppure un ibrido ottenuto sovrapponendo i due tipi. Bene, possiamo già arricchire la nostra mappa fornendo sia la barra di navigazione, aggiungendo la riga map.addControl(new GLargeMapControl());, che i pulsanti per selezionare le diverse modalità di visualizzazione, tramite la riga map.addControl(new GMapTypeControl());
Il codice della funzione verrà quindi modificato come segue

```
var map = new GMap2(document.getElementById("map"));
map.setCenter(new GLatLng(37.4419, -122.1419), 13);
map.addControl(new GLargeMapControl()); // aggiunge barra navigazione
map.addControl(new GMapTypeControl()); //
                                        aggiunge scelta visualizzazione
```

Il problema dei test locali

Se il server risiede sulla propria macchina di sviluppo a cui non è assegnato un dominio, si potrebbe avere il problema che il servizio di Google non riconosce il dominio per cui la chiave è stata registrata. La soluzione più semplice è quella di creare un alias, ovvero un nome simbolico che viene "tradotto" in un indirizzo IP specifico. Per crearlo editare il file hosts e usare un alias a propria scelta che poi verrà usato per generare la chiave di Google. In Windows XP il file da editare è C:\WINDOWS\system32\drivers\etc\hosts, in un sistema GNU/Linux il file è posizionato in /etc/hosts per i test Java è stato creato l'alias sitodiprova.it per l'indirizzo IP locale, ovvero 127.0.0.1 (Figura 3.4):

```
127.0.0.1       localhost sitodiprova.it
```

Ora è possibile ottenere la chiave di Google per l'indirizzo sitodiprova.it e usarla per i propri test locali!

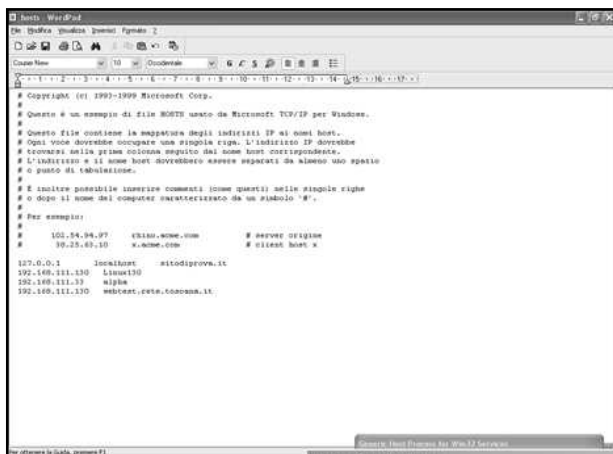
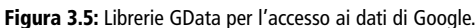



Figura 3.4: Google non riconosce il server locale come corretto? Usare un alias!

DATI DA GOOGLE? GDATA!

Google offre numerosi altri servizi oltre le Google Maps. Essi spaziano dal servizio Blogger (per la creazione di blog) a Calendar (calendario personale con possibilità di condivisione tra più persone) ad applicazioni di produttività indivisuale (spreadsheet ed editor). Un'interessante caratteristiche di tutte queste applicazioni è che possiedono delle API il cui modello di interscambio è comune. Tale modello è chiamato GData (Google Data), e le API sono descritte alla pagina http://code.google.com/apis/gdata/index.html (Figura 3.5). Oltre alle API Google pubblica anche librerie pronte per essere usate nei più comuni linguaggi di programmazione, tra cui Java (versione 1.5), .NET, PHP, Python e Objective-C. Sta crescendo il supporto anche per il linguaggio JavaScript che, per ora, include la possibilità di usare le applicazioni Blogger, Calendar, Spreadsheet e Google Base.
Negli esempi a seguire si mostrerà, tra le altre cose, alcune applicazioni che fanno uso di Calendar.

Figura 3.5: Librerie GData per l'accesso ai dati di Google.

IL PRIMO MASHUP

In questo capitolo si vedrà come realizzare un primo mashup. L'esempio verrà analizzato e implementato sia in Java che PHP (di seguito non ci sarà più questa ambivalenza: via via verranno presentati mashup realizzati o con una o con l'altra tecnologia).

IL PROBLEMA

Si vorrebbe far sì che per ogni visitatore del sito appaia una mappa centrata sul posto da dove il visitatore proviene. Per provenienza si intende la localizzazione della connessione ad Internet e non le impostazioni del browser (o sistema operativo) per la lingua. Un simile problema potrebbe essere affrontato ogni qualvolta si vogliano offrire delle informazioni personalizzate (turistiche, commerciali o semplicemente informative) in base al riconoscimento della provenienza della sua connessione. Così se uno utilizza il proprio portatile da casa, si vedrà recapitare informazioni vicine alla sua casa, se lo utilizza all'estero (per lavoro o vacanza) le informazioni del sito si riferiranno alla sua nuova locazione.

Per la creazione della mappa si può scegliere tra Google Maps o Yahoo Maps. Nell'esempio si sceglierà Google Maps, le cui API sono state descritte in precedenza.

LOCALIZZARE UN INDIRIZZO IP

Tra i diversi servizi a disposizione due sono particolarmente indicati per risolvere il nostro problema: uno è il servizio (gratuito) fornito come CGI da http://api.hostip.info. Il servizio di interesse risponde alla pagina http://api.hostip.info/get_html.php e si possono passare due parametri. Il primo (ip) indica l'ip da analizzare, il secondo (position) indica se si vuole, tra i risultati, anche la latitudine e la longitudine della localizzazione. Per esempio si provi ad aprire il seguente indirizzo con un browser:

http://api.hostip.info/get_html.php?position=true&ip=71.170.59.170

la pagina risultante fornisce le informazioni di georeferenziazione cercate, come mostrato in Figura 4.1

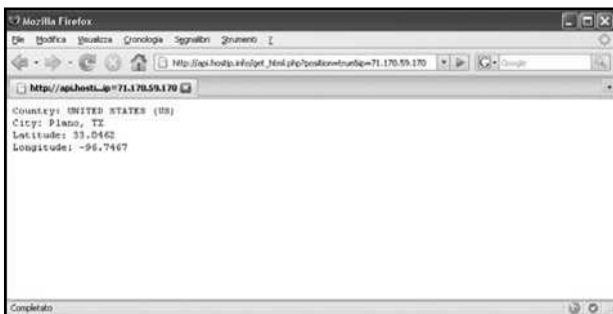



Figura 4.1: le informazioni relative all'indirizzo ip di esempio (71.170.59.170).

Dopo alcune prove sul servizio si può notare che molti indirizzi italiani non sono risolti correttamente (mentre lo sono quasi tutti quelli statunitensi). Ricercando servizi alternativi si può trovare che su http://www.maxmind.com/ c'è GeoIP, un database da scaricare e usare in locale all'applicazione e che permette di risalire alla localizzazione degli IP. Non solo: sul sito sono presenti anche le classi (disponibili in diversi linguaggi tra cui Java e PHP) per leggere i contenuti della base dati. Le prove hanno evidenziato che questa soluzione permette di risolvere anche gran parte degli indirizzi ip provenienti dall'Italia. Pertanto si farà uso anche di tale servizio (confrontando i risultati ottenuti con il servizio precedente, in termini di performance ma anche di altre caratteristiche meno ovvie (costi di sviluppo, manutenzione, affidabilità e così via).

REALIZZARE L'APPLICAZIONE

Essendo il primo servizio realizzato, verrà mostrata sia una imple-

mentazione in Java che una in PHP. Nel seguito si sceglierà una delle due tecnologie per realizzare gli altri mashup.

L'applicazione Java

L'applicazione Java si compone di due parti: una di reperimento delle informazioni ed una di visualizzazione. La prima consiste nel realizzare una o più classi che, opportunamente configurate, permettono di restituire le informazioni di georeferenziazione. La seconda parte si compone di pagine Jsp ed eventuali pagine di supporto (CSS o JavaScript).

Attenzione

Da qui in seguito si userà un particolare package per includere le classi sviluppate. Tale package inizia sempre con it.ioprogrammo.mashup e pertanto non verrà esplicitato. Talvolta si useranno package più specifici; solo in ques'ultimo caso si farà riferimento al package usato.

Una classe per i dati: InfoPoint

La prima classe che si descrive è InfoPoint. Essa non è altro che un Java Bean con le seguenti proprietà: città, stato, latitudine, longitudine e tempoGenerazione. Le prime quattro sono le proprietà utili per la georeferenziazione, l'ultima è solo un'informazione che sarà utile per delle statistiche sui tempi di generazione dei dati dal server. Un JavaBean non è altro che una classe Java con tutte le proprietà private e i metodi per leggere/scrivere tali proprietà che hanno la convenzione sul nome per cui il metodo che legge la proprietà pippo è chiamato getPippo, quello che la scrive setPippo (vista la semplicità della classe InfoPoint essa è presente nei sorgenti allegati e non riportata nel libro; lo stesso sarà fatto per le classi concettualmente banali e che non richiedono ulteriori spiegazioni).

Reperire i dati: LocalizzaIp

La classe LocalizzaIp.java avrà il compito di risolvere, usando un servizio esterno, la georeferenziazione di un indirizzo IP. Il primo metodo che si analizza è infoUrl: esso si occupa di invocare la pagina http://api.hostip.info/get_html.php con gli opportuni parametri ed eseguire il parsing del risultato:

```
public InfoPoint infoUrl(String ip){
  long tempo = (new java.util.Date()).getTime();
  InfoPoint res = null;
  try {
    if (ip.equals("127.0.0.1"))
      ip=ipTest;
    res = new InfoPoint();
    URL url = new URL(urlServizio + ip);
    BufferedReader instr =
      new BufferedReader(
       new InputStreamReader(url.openStream()) );
    String rigaLetta;
    int righe = 0;
    while (righe<4){
      rigaLetta = instr.readLine();
      System.out.println(righe+"> '"+rigaLetta+"'");
      parsingRiga(righe, rigaLetta, res);
      righe++;
    }
  } catch (Exception e) {
    e.printStackTrace();
  } finally{
    if (res!=null)
      res.setTempoGenerazione(
       (new java.util.Date()).getTime() - tempo);
  }
```

```
  return res;
}
```

Si noti che nel caso in cui l'IP sia 127.0.0.1 (ovvero il test viene fatto in locale) si usa un IP predefinito.
Il parsing è piuttosto semplice, visto che il risultato è dato da più righe di testo (dove ciascuna riga è un'informazione ed è formata da etichetta:valore):

```
private void parsingRiga(int righe, String rigaLetta, InfoPoint res){
  String dato = rigaLetta.substring(rigaLetta.indexOf(":")+1);
  double val = 0;
  switch(righe){
    case 0:
      res.setStato(dato);
      break;
    case 1:
      res.setCitta(dato);
      break;
    case 2:
      try{
        val = Double.parseDouble(dato);
      }catch(NumberFormatException e){;}
      res.setLatitudine(""+val);
      break;
    case 3:
      try{
        val = Double.parseDouble(dato);
      }catch(NumberFormatException e){;  }
      res.setLongitudine(""+val);
  }
}
```

Ecco invece la funzione che georeferenzia un indirizzo IP usando il database GeoIP:

```
public  InfoPoint infoGeo(String ip){
  long tempo = (new java.util.Date()).getTime();
  InfoPoint res=null;
  try{
    if (ip.equals("127.0.0.1"))
      ip = ipTest;
    Location l1 = cl.getLocation(ip);
    if (l1!=null){
      InfoPoint infop = new InfoPoint();
      infop.setCitta( l1.city );
      infop.setStato(l1.countryName);
      infop.setLatitudine(""+l1.latitude);
      infop.setLongitudine(""+l1.longitude);
      res=infop;
    }
  } finally{
    if (res!=null)
      res.setTempoGenerazione(
       (new java.util.Date()).getTime() - tempo);
  }
  return res;
}
```

Come si vede la classe fa uso di cl, la quale è un attributo della classe così dichiarato:

```
private static LookupService cl = null;
```

Inoltre esso va inizializzato; in particolare questo avviene nel costruttore della classe:

```
private LocalizzaIp(){
  try{
    cl = new LookupService(pathGeoLiteCity,
      LookupService.GEOIP_MEMORY_CACHE
      | LookupService.GEOIP_CHECK_CACHE );
  }catch(Exception ex){
    ex.printStackTrace();
  }
}
```

La classe LookupService, insieme ad altre classi per l'accesso alla base dati degli indirizzi, è scaricabile dalla pagina Web http://www.maxmind.com/app/api (far riferimento alla documentazione inclusa per l'utilizzo delle API fornite).

La parte di visualizzazione

Ottenute le informazioni, in un oggetto di tipo InfoPoint, è necessario creare una Google Map che mostri un marcatore sulla posizione data dalla longitudine e latitudine calcolati, nonché centrata sul punto stesso. La parte di visualizzazione si compone di due file: localizzaIp.jsp, che contiene il codice html con la mappa e la dichiarazione delle funzioni JavaScript da usare, e dati-locale.jsp, che è un file contenente solo codice JavaScript ma con una parte dinamica (fatta in JSP). Il primo file si compone, tra le altre cose, di due placeholder per le due mappe: una che visualizzerà la mappa con le coordinate ottenute dal metodo infoUrl (che fa uso del servizio remoto hostInfo), l'altra che visualizzerà la mappa con le coordinate ottenute dal database locale GeoIp:

```
<div id="map1" class="map1"></div>
<div id="map2" class="map1"></div>
```

Tra il codice JavaScript della pagina localizzaIp ci sarà anche quello

che si occupa dell'inizializzazione:

```
<script type = "text/javascript" src="./dati-locale.jsp"> </script>
<script type="text/javascript">

function load() {
 if (GBrowserIsCompatible()) {
   var map1 = new GMap2(document.getElementById("map1"));
   disegnaPrima(map1);
   var map2 = new GMap2(document.getElementById("map2"));
   disegnaSeconda(map2);
 }
}
</script>
```

Nel file dati-locale.jsp troverà posto il seguente codice:

```
String host = request.getRemoteHost();
```

Nella stringa host sarà contenuto l'Ip del visitatore. Da tale indirizzo ecco come ottenere la referenziazione usando GeoIp:

```
LocalizzaIp loc= LocalizzaIp.getInstance();
InfoPoint info = loc.infoGeo(host);
```

Ad esso corrisponderà la seguente funzione JavaScript per l'inizializzazione della mappa:

```
function disegnaPrima(mappa) {
 var punti = new Array();
 var gtext = new Array();

 mappa.setCenter(
```

```
 new GLatLng(
  <%= info.getLatitudine() %>,
  <%= info.getLongitudine() %>),
 8);
punti [0]=
 new GLatLng (
  <%= info.getLatitudine() %>,
  <%= info.getLongitudine() %>);
gtext [0]= <%= dammiDescrizione(host, info) %>
disegnaMappa(mappa, punti, gtext);
}
```

In maniera simile ecco il reperimento delle informazioni dal servizio hostInfo:

```
<%
 info = loc.infoUrl(host);
%>
```

e il disegno della relativa mappa:

```
function disegnaSeconda(mappa) {
 var punti = new Array();
 var gtext = new Array();

 mappa.setCenter(
  new GLatLng(
   <%= info.getLatitudine() %>,
    <%= info.getLongitudine() %>),
    8);
 punti [0]= new GLatLng (
  <%= info.getLatitudine() %>,
   <%= info.getLongitudine() %>
```

```
 );
 gtext [0]= <%= dammiDescrizione(host, info) %>
 disegnaMappa(mappa, punti, gtext);
}
```

Entrambe le funzioni JavaScript fanno uso della seguente funzione:

```
function disegnaMappa(mappa, punti, testo) {
  mappa.addControl(new GLargeMapControl());
  mappa.addControl(new GMapTypeControl());
  for (i=0; i<punti.length; i++){
      mappa.addOverlay (creaFumetto(punti[i],testo[i]));
  }
}
```

In Figura 4.2 e 4.3 due esempi di indirizzi IP (nel primo caso risolti correttamente da entrambi i servizi, nel secondo caso no!).

Figura 4.2: Un indirizzo IP (che vale 82.51.68.55) risolto correttamente da entrambi i servizi.

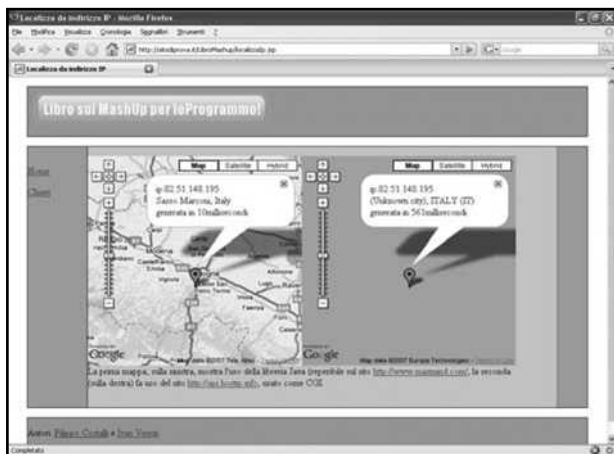



Figura 4.3: Un indirizzo IP (che vale 82.51.148.195) risolto correttamente solo da GeoIP.

L'APPLICAZIONE PHP

Cerchiamo di applicare anche alla parte PHP il pattern MVC ed una gestione ad oggetti. Il paradigma della programmazione orientata agli oggetti è stato introdotto con la versione 5 di PHP, in cui i progettisti hanno cercato di inserire concetti similjava. È possibile, quindi, sforzarsi per cercare di dare al nostro codice una struttura object-oriented e per disaccoppiare le componenti riguardanti logica dell'elaborazione da quelle incaricate della visualizzazione dei risultati. In questo modo riusciremo ad ottenere codice con una migliore leggibilità e più facilmente modificabile.
Il concetto di georeferenziazione resta, ovviamente, immutato rispetto a quanto introdotto in precedenza, in quanto indipendente dal linguggio utilizzato.

Una classe per i dati: infopoint

Anche nell'esempio php utilizzeremo una classe, analoga alla Info-

Point dell'esempio Java, i cui attributi saranno, anche in questo caso, i seguenti: città, stato, latitudine, longitudine e tempoGenerazione.
Una classe php è una struttura contenente una serie di attributi che possono essere riferiti con la notazione

```
$nomeoggetto->nomeattributo
```

Il codice della classe infopoint (contenuto nel file infopoint.php) è estremamente semplice ed è il seguente:

```
class infopoint {
  var $citta;
  var $stato;
  var $latitudine;
  var $longitudine;
}
```

Reperimento dei dati

Andiamo ad illustrare l'implementazione in PHP della funzione che georeferenzia un indirizzo IP tramite l'utilizzo del database GeoIP

```
<?php
include("geoipcity.inc");
include("infopoint.php");

function infoGeo($ip){
  $gi = geoip_open("GeoLiteCity.dat", GEOIP_STANDARD);
  $record = geoip_record_by_addr($gi, $ip);

  $infop = new infopoint();
  $infop->latitudine = $record->latitude;
  $infop->longitudine = $record->longitude;
```

```
  $infop->citta = $record->city;
  $infop->stato = $record->country_name;

  geoip_close($gi);
  return $infop;
}
?>
```

La funzione accetta come parametro un indirizzo ip e restituisce un oggetto di tipo infopoint. Analizziamone sinteticamente il codice. La prima istruzione restituisce, di fatto, una connessione al database GeoLiteCity.dat, la seconda ritorna un oggetto $record di classe dal quale estraiamo gli attributi che ci interessano per assegnarli ad un oggetto infopoint. Una volta chiusa la connessione viene ritornato l'oggetto infopoint.
Attenzione: in questo esempio si suppone che il file GeoLiteCity.dat sia posizionato nella stessa directory della pagina PHP. Se questo non fosse vero occorrerà settare il path relativo adeguato per accedervi.
La funzione successiva si occupa di risolvere la georeferenziazione dell'indirizzo IP che riceve come parametro, mediante l'utilizzo del servizio infoUrl.

```
function infoUrl($ip){

  $record = file('http://api.hostip.info/get_html.php?ip='
              . $ip.'&position=true');

  $infop = new infopoint();
  $infop->latitudine = substr($record[2], 11);
  $infop->longitudine = substr($record[3], 12);
  $infop->citta = substr($record[1], 8);
  $infop->stato = $record->country_name;
```

```
  geoip_close($gi);

  return $infop;
}
```

Componente di visualizzazione

Anche nell'implementazione PHP occorrerà, a partire dalla longitudine e dalla latitudine fornite dall'oggetto infopoint, generare una Google Map centrata sul punto così descritto.
Analogamente a quanto visto in precedenza viene disaccoppiata la parte html contenente la mappa da quella riguardante le funzioni Javascript.
La parte di visualizzazione non si differenzia molto, concettualmente, da quanto visto nell'esempio Java. Le differenze, essenzialmente, riguardano la parte del codice generata dinamicamente, ma sono tali solo per la differente sintassi del PHP rispetto a JSP.
La parte contenente la mappa è pressochè identica, eccezion fatta per la sintassi con cui vengono inclusi gli header ed i footer della pagina e per i riferimenti alla pagine javascript.

```
<script type = "text/javascript" src="./datilocale.php"> </script>
```

Nel file datilocale.php si trova il seguente codice che assegna alla variabile denominata host l'ip del visitatore.

```
$host = getenv("REMOTE_ADDR");
```

Una volta ottenuto l'indirizzo del client viene invocata la funzione infogeo che, tramite l'utilizzo di GeoIP, ritorna le coordinate geografiche relative all'ip chiamante.

```
$info =  infoGeo($host);
```

La funzione JavaScript per l'inizializzazione della mappa corrispondente differisce in maniera minima rispetto alla sua analoga in ambiente JSP. Le differenze, come accennato in precedenza, riguardano solo la differente sintassi per richiamare dinamicamente la latitudine e la longitudine ottenute:

```
function disegnaPrima(mappa) {
 var punti = new Array();
 var gtext = new Array();

 mappa.setCenter(
  new GLatLng(
   <?= $info->latitudine ?>,
   <?= $info->longitudine ?>),
  8);
 punti [0]=
  new GLatLng (
   <?= $info->latitudine ?>,
   <?= $info->latitudine ?>);
 gtext [0]= <?= dammi_descrizione($host, $info) ?>
 disegnaMappa(mappa, punti, gtext);
}
```

Per la referenziazione tramite il servizio hostinfo è invece utilizzato il seguente codice:

```
$info =  infoUrl($host);
```

Viene restituita una istanza della classe infopoint di cui vengono utilizzati gli attributi latitudine e longitudine per generare dinamicamente la funzione javascript di disegno della mappa:

```
function disegnaSeconda(mappa) {
```

```
var punti = new Array();
var gtext = new Array();

mappa.setCenter(
  new GLatLng(
  <?= $info->latitudine ?>,
  <?= $info->longitudine ?>),
    8);
punti [0]= new GLatLng (
  <?= $info->latitudine ?>,
  <?= $info->longitudine ?>),
 );
gtext [0]= <?= dammi_descrizione($host, $info) ?>
disegnaMappa(mappa, punti, gtext);
}
```

Allo stesso modo dell'esempio Java ecco, di seguito, il risultato dei due esempi sviluppati in PHP: il primo con un indirizzo che entrambi i servizi sono riusciti a risolvere, il secondo con un indirizzo che solamente GeoIP è riuscito a referenziare (Figure 4.4 e 4.5)

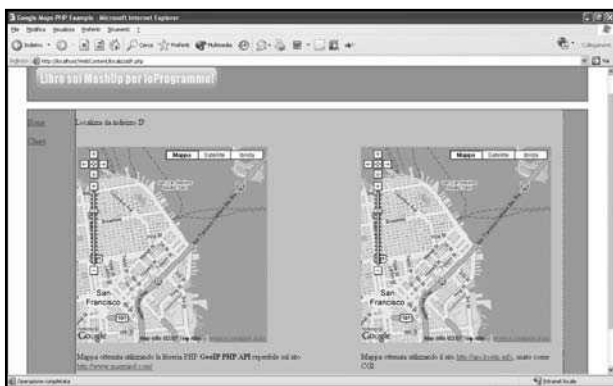



Figura 4.4: Un indirizzo IP risolto correttamente da entrambi i servizi.

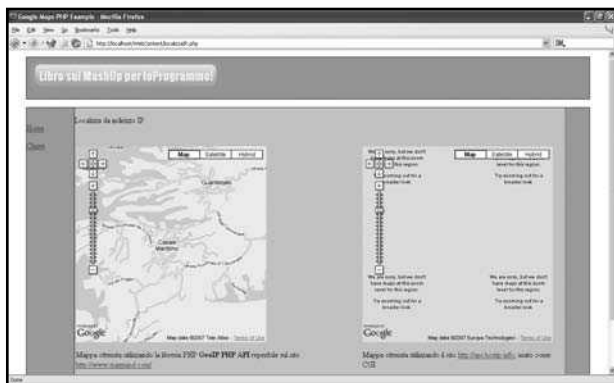



Figura 4.5: Un indirizzo IP risolto correttamente solo da GeoIP.

CONSIDERAZIONI SULL'ESEMPIO

L'esempio che fa uso del CGI per reperire le informazioni relative alla localizzazione dell'indirizzo IP e l'uso di Google Maps per la sua visualizzazione è senz'altro un esempio di mashup (infatti si usano due servizi separati, ciascuno acceduto sul Web). Si può dire la stessa cosa nel caso in cui la localizzazione è presa da una base dati locale? Probabilmente no; però si vede subito la difficoltà nel definire il termine mashup. È chiaro che per l'utente finale in fondo il risultato è lo stesso. Inoltre, anche nel primo esempio, si può esservare come le informazioni reperite in rete siano "risolte" in due momenti diversi: il CGI viene invocato lato server (via codice Java o PHP) mentre le mappe vengono invocate lato client (via codice JavaScript). È un classico esempio di mashup ibrido. Altri contesti potrebbero prevedere mashup solo lato server o solo lato client. Quali preferire? Ovviamente non c'è (come spesso accade) una risposta definitiva, ma ciascuna soluzione presenta alcuni vantaggi e alcuni aspetti critici…

Mashup lato client

Un mashup solo lato client ha il vantaggio di non appesantire il ser-

ver di ulteriori elaborazioni. Questo può essere particolarmente importante per le applicazioni a cui accedono molti utenti. Inoltre è indubbio che ciascun utente vedrà sempre le informazioni aggiornate non appena queste vengono messe a disposizione dai siti che fungono da sorgente dati. Il problema principale è realizzare applicazioni davvero cross browser, ovvero che funzionino con i diversi browser presenti sul mercato. Questa caratteristica fa sì che è preferibile usare librerie Ajax piuttosto che pensare di realizzarne di proprie ad hoc. Infatti lo sviluppo di applicazioni cross browser è un'attività piuttosto specifica e complessa, sia per il bagaglio di conoscenze richieste, sia per la difficoltà di test esaustivi.

Mashup lato server

Fornire dei mashup risolti già a livello server porta a "spendere" più risorse sul server dell'applicazione, ma questa può avvalersi di forme di cache dei dati che può portare a notevoli vantaggi sia in termini di performance (in quanto i dati sono reperiti da una cache locale e non sempre da remoto; in tal senso si confrontino i tempi di "risoluzione" degli indirizzi IP dell'esempio realizzato: accedere ad una risorsa attraverso una chiamata http è quasi sempre più lento che accedere ad una base dati locale) ma anche di evitare che i propri utenti intasino di richieste i server che fungono da sorgenti di dati (spesso questa caratteristiche è una richiesta fatta da server che espongono i dati!). Per questo motivo la gran parte dei mashup, quando possibile, viene risolta direttamente lato server.

MAPPE CON IL METEO E GEOCODING

In Il senso di un'applicazione di mashup che utilizzi le mappe di Google è quello di disegnare una mappa mettendo in relazione le informazioni che vogliamo rappresentare con una latitudine ed una longitudine. Vi sono numerosi servizi di geocoding che, a partire da un indirizzo postale, restituiscono con precisione le coordinate geografiche corrispondenti. Nel corso di questo capitolo verrà illustrata un'applicazione che, acquisendo come parametro di ingresso un indirizzo postale, visualizzerà la mappa centrata su tale indirizzo e le informazioni meteorologiche relative all'area geografica (la provincia) richiesta. Per il mashup verranno utilizzati:

- Il servizio web di geocoding di Google
- Google Maps
- Il servizio metereologico weather.com

GOOGLE MAPS API GEOCODER

Un servizio di geocoding, abbiamo detto, restituisce la latitudine e la longitudine corrispondenti ad un indirizzo postale inviato come parametro. Le informazioni restuite dai servizi di geocoding sono in formato XML, ed ogni servizio restituisce un formato proprio. Dunque per usufruire di questi servizi occorre connettersi ad essi, per poi ottenere una risposta in formato XML che andrà poi parserizzata per ottenere le informazioni (latitudine e longitudine) di cui necessitiamo per la nostra applicazione.
Lo strumento di geocoding analizzato sarà il Google Maps API geocoder, accessibile all'indirizzo http://www.google.com/apis/maps/documentation/#Geocoding_Examples. Google assicura che questo servizio è in grado di garantire un dettaglio molto preciso per i seguenti paesi: Stati Uniti, Canada, Francia, Italia, Germania e Spagna.

Il formato di Request e Response

Il servizio accetta come parametro un indirizzo postale in un particolare formato, detto REST (acronimo per REpresentional State Transfer). Si tratta, di fatto, di una richiesta GET del protocollo HTTP, quindi di una stringa composta da un URL seguito da un elenco di parametri url?chiave1=velore1&chiave2=valore2...
Una particolarità estremamente interessante è che Google non impone particolari rigidità sulla formattazione dell'indirizzo inviato come parametro. Infatti non sono richiesti particolari campi separatori tra il nome della via, il numero, la città, la nazione ed il codice di avviamento postale. Il servizio analizza quanto gli viene fornito in ingresso e prova a dedurne la risposta migliore da fornire. La risposta che fornisce, contenente la latitudine e la longitudine ricavate dall'indirizzo, è invece strutturata in maniera ben precisa attraverso l'utilizzo di un dialetto di XML detto KML (Keyhole Markup Language). Per esempio, una semplice richiesta di questo tipo:

```
http://maps.google.com/maps/geo?q=Via+Corridoni+12+56100+
                                             Pisa+PI+Italia
```

Otterrà una risposta in formato XML analoga alla seguente:

```
<kml xmlns="http://earth.google.com/kml/2.0">
 <Response>
   <name>Via Corridoni 12 56100 Pisa PI Italia</name>
   <Status>
     <code>200</code>
     <request>geocode</request>
   </Status>
   <Placemark id="p1">
     <address>Via Filippo Corridoni, 12, 56125 Pisa, PI (Toscana),
                                                    Italy</address>
     <AddressDetails Accuracy="8" xmlns="urn:oasis:names:tc:ciq:
```

```
                                                    xsdschema:xAL:2.0">
    <Country>
      <CountryNameCode>IT</CountryNameCode>
      <AdministrativeArea>
        <AdministrativeAreaName>Toscana</AdministrativeAreaName>
        <SubAdministrativeArea>
        <SubAdministrativeAreaName>PI</SubAdministrativeAreaName>
          <Locality>
            <LocalityName>Pisa</LocalityName>
            <Thoroughfare>
              <ThoroughfareName>Via Filippo Corridoni,
                                              12</ThoroughfareName>
            </Thoroughfare>
            <PostalCode>
              <PostalCodeNumber>56125</PostalCodeNumber>
            </PostalCode>
          </Locality>
        </SubAdministrativeArea>
      </AdministrativeArea>
    </Country>
   </AddressDetails>
   <Point>
    <coordinates>10.406862,43.706173,0</coordinates>
   </Point>
  </Placemark>
 </Response>
</kml>
```

Il messaggio ritornato è contenuto nel tag Response suddiviso in tre entità principali:

- name: Contiene il parametro inviato al geocoder
- Status: Il codice di risposta, nell'esempio il codice è 200, che sta

a significare che la richiesta è stata processata correttamente e che è stata fornita una risposta.

- PlaceMark: questo tag è presente solo se la richiesta è andata a buon fine e contiene tutte le indicazioni richieste, in particolare contiene l'entità Point (evidenziata in grassetto) all'interno della quale sono presenti le coordinate geografiche corrispondenti all'indirizzo postale richiesto.

Analizzati il formato delle richieste da inviare al Geocoder Google e delle conseguenti risposte ottenute, occorre costruire una classe che invii una richiesta ed elabori i dati ottenuti in risposta.
Per richiedere il servizio web utilizzeremo il package CURL introdotto nel Capitolo 2, mentre per la parserizzazione del dato in formato KML faremo ricorso alla libreria SimpleXML il cui funzionamento è stato illustrato sempre nel Capitolo 2.

La funzione che utilizza il servizio

Il codice che segue illustra la funzione PHP che utilizza CURL e SimpleXML per ritornare un oggetto di tipo infopoint a fronte di un indirizzo postale passato come parametro.

```
function infoAddress($postal_address)
{
  $infop = new infopoint();

// Creazione oggetto CURL
  $curl_obj = curl_init();
  curl_setopt($curl_obj, CURLOPT_HEADER, 0);
  curl_setopt($curl_obj, CURLOPT_RETURNTRANSFER, 1);

  // Carica la chiave Google salvata nel file ApiKey.txt
  $fh = fopen("ApiKey.txt", "r");
if (!$fh) die("Errore: Chiave per le API Google non trovata ");
```

```
$api_key = fread($fh, filesize("ApiKey.txt"));
fclose($fh);
// Preparazione della stringa di richiesta
  $url = "http://maps.google.com/maps/geo?output=xml";
  $url .= "&key=$api_key";
  $url  .= "&q=$postall_address";
  // Effettua la query a Google
  curl_setopt($curl_obj, CURLOPT_URL, $url);
  $geo_response = curl_exrc($curl_obj);
  // Parsing della risposta
  $geo_result = simplexml_load_string($geo_response);
  $response_status = $geo_result->Response->Status->code;
  if ($response_status != 200)
  {
    echo("Impossibile soddisfare richiesta per questo indirizzo";  }
  else
  {
    // Estrae le informazioni e le assegna all'oggetto infopoint
    $coord = $geo_result->Response->Placemark->Point->coordinates;
    $list($lat, $lon) = split(",", $coord);
    $citta= $geo_result->Response->Placemark->Country
    ->AdministrativeArea-> SubAdministrativeAreaName;
    $infop->latitudine = $lat;
    $infop->longitudine = $lon;
    $infop->citta = $citta;
  }
  curl_close($curl_object);
return $infop ;
}
```

La funzione instanzia un oggetto CURL e gli fornisce la richiesta REST che viene trasmessa a Google e la cui risposta è contenuta nell'oggetto $geo_response che – abbiamo detto – è nel formato KML.

Utilizzando semplicemente l'istruzione simplexml_load_string, fornita dalla libreria Simple XML, tutte le informazioni contenute nel documento KML vengono trasferite all'interno di una gerarchia di oggetti, facilmente navigabile, la cui radice è nell'oggetto $geo_result.. L'estrazione della latitudine e della longitudine consiste semplicemente nel navigare l'oggetto sino al livello dell'entità coordinates. Con la stessa metodologia estraiamo la provincia che ci servirà per richiedere le relative previsioni meteo. Tali operazioni sono evidenziate in grassetto nel codice precedente.
La funzione così descritta può essere utilizzata ogni volta che si desideri reperire le coordinate geografiche di un determinato indirizzo postale. Per utilizzarla è necessario essere provvisti di una apposita Google key per il dominio utilizzato che, nell'esempio, abbiamo supposto essere salvata in un file denominato ApiKey.txt e posto al medesimo livello di filesystem del file PHP contenente il codice.
Occorre adesso utilizzare le coordinate ottenute per mostrare la mappa - centrata sul punto che esse descrivono – e le informazioni metereologiche relative. Per la visualizzazione delle previsioni meteo verrà utilizzato il servizio weather.org con le modalità descritte nel paragrafo seguente.

UN SERVIZIO METEO: WEATHER.COM

Per acquisire informazioni meteorologiche riguardanti una determinata locazione geografica viene utilizzato il servizio fornito dal National Weather Service statunitense, raggiungibile all'indirizzo *http://weather.com*.
Una volta iscritti al sito occorre collegarsi all'indirizzo *http://www.weather.com/services/xmloap.html*, per ottenere la chiave di licenza (License key) ed un identificativo (partner id). Oltre a questo verrà fornito un link per scaricare il kit di sviluppo che spiega il formato XML adottato nonché le icone da utilizzare. Nell'esem-

pio in oggetto le icone, una volta scaricato il kit, sono state salvate nella directory <APP_ROOT>/img, serviranno per mostrare la situazione metereologica della zona considerata.

Formato delle richieste e delle risposte

A fronte di una richiesta di questo tipo:

http://xoap.weather.com/search/search?where=Pisa

Il servizio risponde con un documento del del tipo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<search ver="2.0">
  <loc id="ITXX0059" type="1">Pisa, Italy</loc>
</search>
```

Effettuando il parsing del documento si ottiene, dal tag id, il codice della località richiesta. Ottenuto questo occorre effettuare una nuova richiesta specificando nei parametri l'identificativo della località, il partner id e la chiave di licenza ottenuti in precedenza. Il formato della richiesta è il seguente, in grassetto i parametri inseriti.

```
http://xoap.weather.com/weather/local/ITXX0059
            ?cc=*&prod=xoap&par=1043388194&key=aee01957b2387dae
```

Weather.com fornisce la seguente risposta:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<weather ver="2.0">
  <head>
    ...
  </head>
  <loc id="ITXX0059">
```

```
  <dnam>Pisa, Italy</dnam>
  ...
 </loc>
 <cc>
  <lsup>7/9/07 12:45 PM Local Time</lsup>
  <obst>Pisa, Italy</obst>
  <tmp>81</tmp>
  <flik>81</flik>
  <t>Partly Cloudy</t>
  <icon>30</icon>
  <bar>
   <r>29.91</r>
   <d>steady</d>
  </bar>
  <wind>
   <s>7</s>
   <gust>N/A</gust>
   <d>240</d>
   <t>WSW</t>
  </wind>
  <hmid>48</hmid>
  <vis>6.2</vis>
  <uv>
   <i>N/A</i>
   <t>N/A</t>
  </uv>
  <dewp>59</dewp>
  <moon>
   <icon>24</icon>
   <t>Waning Crescent</t>
  </moon>
 </cc>
</weather>
```

Come si vede la risposta è piuttosto articolata, sono ritornate una molteplicità di informazioni quali la temperatura, la pressione, il vento, la fase lunare ecc... In particolare utilizzeremo la temperatura (espressa in gradi farenheit) e l'icona da utilizzare. Nell'esempio l'icona da utilizzare è la n.30, che andrà reperita con il cammino <APP:_ROOT>/img/30.png.

La funzione implementata

La funzione che utilizza il servizio weather.com riceve come parametro l'oggetto infopoint ottenuto col geocoding e ritorna il codice HTML per la visualizzazione delle caratteristiche meteo.

```
function infoMeteo($citta)
{
// Creazione oggetto CURL
  $curl_obj = curl_init();
  curl_setopt($curl_obj, CURLOPT_HEADER, 0);
  curl_setopt($curl_obj, CURLOPT_RETURNTRANSFER, 1);
  // Carica la chiave weather.com salvata nel file weatherKey.txt
 $fh = fopen("weatherKey.txt", "r");
if (!$fh) die("Errore: Chiave weather.com assente");
$weather_key = fread($fh, filesize("weatherKey.txt"));
fclose($fh);
// Carica il partner id salvata nel file weatherPID.txt
$fh = fopen("weatherPID.txt", "r");
if (!$fh) die("Errore: Partner id weather.com assente");
$weather_pid = fread($fh, filesize("weatherPID.txt"));
fclose($fh);
  // Preparazione della stringa di richiesta
  $url = "http://xoap.weather.com/search/search?where=".$citta;
  // Effettua la query a weather
  curl_setopt($curl_obj, CURLOPT_URL, $url);
  $weather_response = curl_exrc($curl_obj);
```

```
// Parsing della risposta
$weather_result = simplexml_load_string($weather_response);
$city_id = $weather_result->search->loc->attributes["id"];
$html_code="";
if (!$city_id)
{
  echo("Impossibile soddisfare richiesta per questa località";
}
else
{
  // Effettua la richiesta meteo
  $url = "http://xoap.weather.com/weather/local/";
  $url .= $city_id."?cc=*prod=xoap";
  $url .= "&par=".$weather_pid;
  $url .= &key=".$weather_key;

  curl_setopt($curl_obj, CURLOPT_URL, $url);
  $weather_response = curl_exrc($curl_obj);

  $weather_result = simplexml_load_string($weather_response);
  $icon = $weather_result->weather->cc->icon;
  $temp = $weather_result->weather->cc->tmp;

  $html_code="<img src=\"images/".$icon.".png\"/><br/>";
  $html_code="Temperatura:".$temp;
}
curl_close($curl_object);
return $html_code;
}
```

La pagina principale richiamerà quindi inizialmente la funzione

```
$info = infoAddress($postal_address);
```

Passerà poi l'attributo città dell'oggetto ottenuto alla funzione infoMeteo

```
$citta =$info->citta;
$html_temp= infoMeteo($citta);
```

A questo punto visualizzerà la mappa di Google centrata sulle coodinate fornite da infoAddress e inserirà il codice HTML riguardante la temperatura.

Figura 5.1: L'esempio completo in PHP.

MASHUP IN JAVA: CHE TEMPO FA?

Potrebbe essere interessante permettere all'utente di cliccare un punto qualsiasi di una mappa (anche in questo caso realizzata con Google Maps) e fornire le indicazioni meteorologiche (presenti e, se possibile, future) per quella determinata zona. Una veloce ricerca con Google permette di identificare alcuni possibili servizi che forniscono informazioni meteorologiche potrebbe essere riusato, anche per questo esempio, http://www.weather.com. Però si ricercheranno, a titolo di esempio, ulteriori alternative. Un'altra fonte di servizi, tra

cui anche altri diversi da quelli ricercati, è Strike Iron http://www.strikeiron.com/ (in particolare si veda il servizio http://www.strikeiron.com/ProductDetail.aspx?p=247, Figura 5.2, il cui WSDL è localizzato all'indirizzo *http://ws.strikeiron.com/Inner Gears/WeatherByCity2?WSDL*). Nulla di gratuito? Forse…

Figura 5.2: Il servizio (commerciale) fornito da StrikeIron per il tempo meteorologico.

Weather.org

Particolarmente interessante il sito del NOAA (National Oceanic and Atmospheric Administration), agenzia americana che tra i vari servizi offre informazioni meteorologiche sul sito http://www.weather.gov/ (Figura 5.3) in maniera gratuita. L'unico inconveniente è che copre unicamente gli Stati Uniti d'Amarica.

Tra gli altri servizi offre la possibilità di recuperare informazioni meteorologiche in base alle coordinate (longitudine e latitudine) fornite. L'url da invocare è la seguente:

Figura 5.3: Il sito http://www.weather.gov/, con numerose informazioni e statistiche meteorologiche..

http://www.weather.gov/forecasts/xml/OGC_services/ndfdOWSserver.php
I parametri da passare, sempre alla url come query string, sono elencati in Tabella 5.1.

Parametro	Valore usato	Significato
SERVICE	WFS	Nome del servizio richiesto
Request	GetFeature	Tipo di richiesta
Version	1.0.0	Tipo di risultato (diverse versioni producono XML differenti)
TYPENAME	Forecast_Gml2Point	
time	(data attuale)	Tempo (in formato AAAA-MM-GGThh:mm:ss) di cui si vuole la situazione meteorologica
ELEMENTS	maxt,wspd,wdir,rhm, waveh,wx,mint,icons, temp,sky,td,apt,qpf,s now,wgust,pop12	Quali elementi includere nei risultati (ciascun elemento identifica una misurazione; maxt è la temperatura massima, icons è l'icona che illustra la situazione meteorologica e così via).
latLonList	LONG,LAT	Longitudine e latitudine del punto di cui si desiderano le informazioni

Tabella 5.1: Parametri del servizio weather.org

Un esempio potrebbe essere la seguente URL:

http://www.weather.gov/forecasts/xml/OGC_services/ndfdOWSserver.php?SERVICE=WFS&Request=GetFeature&VERSION=1.0.0&TYPENAME=Forecast_Gml2Point&latLonList=43.70,-102.65&time=2007-06-25T23:00:00. La risposta è un documento XML così formato:

```
<?xml version="1.0" encoding="UTF-8"?>
<app:NdfdForecastCollection
xmlns="http://www.weather.gov/forecasts/xml/OGC_services"
xmlns:app="http://www.weather.gov/forecasts/xml/OGC_services"
xmlns:ows="http://www.opengis.net/ows"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.weather.gov/forecasts/xml/OGC_
                                                    services
 http://www.weather.gov/forecasts/xml/OGC_services/schema/dwGML_WF
                                           S_GMLv212.xsd ">

<gml:boundedBy>
  <gml:Box srsName="EPSG:4326">
    <gml:coordinates>-102.65,43.70 -102.65,43.70</gml:coordinates>
  </gml:Box>
</gml:boundedBy>
<gml:featureMember>
 <app:Forecast_Gml2Point>
   <gml:position>
     <gml:Point srsName="EPSG:4326">
       <gml:coordinates>-102.65,43.70</gml:coordinates>
     </gml:Point>
```

```
    </gml:position>
    <app:validTime>2007-06-25T23:00:00</app:validTime>
    <app:maximumTemperature>101</app:maximumTemperature>
    <app:minimumTemperature>60</app:minimumTemperature>
    <app:temperature>98</app:temperature>
    <app:dewpointTemperature>53</app:dewpointTemperature>
    <app:apparentTemperature>95</app:apparentTemperature>
    <app:rainAmount6Hourly>0.00</app:rainAmount6Hourly>
    <app:snowAmount6Hourly>0</app:snowAmount6Hourly>
    <app:probOfPrecip12hourly>13</app:probOfPrecip12hourly>
    <app:windSpeed>7</app:windSpeed>
    <app:windGust>8</app:windGust>
    <app:windDirection>257</app:windDirection>
    <app:skyCover>38</app:skyCover>
    <app:relativeHumidity>23</app:relativeHumidity>
    <app:waveHeight>9999</app:waveHeight>
    <app:weatherCoverage>none</app:weatherCoverage>
    <app:weatherIntensity>none</app:weatherIntensity>
    <app:weatherType>none</app:weatherType>
    <app:weatherQualifier>none</app:weatherQualifier>
    <app:weatherVisibility>none</app:weatherVisibility>
<app:weatherIcon>http://www.nws.noaa.gov/weather/images/fcicons/hot.
                                        jpg</app:weatherIcon>
   </app:Forecast_Gml2Point>
</gml:featureMember>
</app:NdfdForecastCollection>
```

Utilizzeremo questo servizio per il nostro mashup di esempio.

Invocazione e uso del servizio

La classe che si occupa di gestire le invocazioni al servizio è FiltroSitoWeatherOrg.java. Tra le altre cose essa si occupa anche del parsing del documento XML restituito. Per comodità implementativa

questa classe non restituisce una classe JavaBean con i valori desiderati, ma una hashtable (chiave/valore). La classe dichiara alcune costanti (endpoint del servizio e valori di invocazione fissi) e identifica in tagsWeather tutti i tag di interesse (di cui cioè si reperisce il valore):

```
public static final String[] tagsWeather = {
  "temperature",  "weatherIcon", "maximumTemperature",
  "minimumTemperature", "rainAmount6Hourly", "snowAmount6Hourly",
  "probOfPrecip12hourly", "skyCover" , "relativeHumidity",
  "windSpeed"
};
```

Il metodo principale è prendiCondizioniMeteo che, date longitudine e latitudine, restituisce una Hashtable con tutti i valori reperiti dal servizio remoto:

```
public static Hashtable prendiCondizioniMeteo(String lat, String lon) {
  try{
      String urlComp = urlBase+"?"+
        paramsBase+
        "&latLonList="+lat+","+lon+
        "&time="+dateService.format(new java.util.Date())+
        "&ELEMENTS=maxt,wspd,wdir,rhm,waveh,wx,mint,"+
        "icons,temp,sky,td,apt,qpf,snow,wgust,pop12";
      java.net.URL url = new java.net.URL( urlComp );
      return prendiAttributi(url.openStream());
  }catch(Exception ex){
      ex.printStackTrace();
      return new java.util.Hashtable();
  }
}
```

Ed ecco il metodo che si preoccupa di fare il parsing del risultato, usando le primitive Java per la gestione di documenti XML:

```
public static Hashtable prendiAttributi(java.io.InputStream input) {
  Hashtable result = new Hashtable ();
  try {
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    DocumentBuilder db = dbf.newDocumentBuilder();
    Document dom = db.parse(input);
    Element docEle = dom.getDocumentElement();
    NodeList nl =
      docEle.getElementsByTagName("app:Forecast_Gml2Point");
    if (nl != null && nl.getLength() > 0) {
      for (int i = 0; i < nl.getLength(); i++) {
        Element el = (Element) nl.item(i);
        for(int ix=0; ix<tagsWeather.length; ix++)
          result.put(
           tagsWeather[ix],
           getValue(el, tagsPrefix+tagsWeather[ix])
          );
      }
    }else
      System.out.println("Nessun elemento app:Forecast_Gml2Point!!");
  } catch (Exception e) {
    e.printStackTrace();
  }
  return result;
}
```

Tale parsing consiste nella costruzione del DOM (Document Object Model) ovvero di tutta la struttura del docuemnto XML come oggetto in memoria. Questo può essere sensato per messaggi di risposta il cui contenuto è, in fondo, così semplice. Nel caso di documen-

ti molto più lunghi e complessi è preferibile ricorrere a tecnologie alternative, quali SAX e StAX, che evitano di costruire oggetti in memoria ma funzionano in maniera più snella.

Sul Web
Per approfondire le tecnologie di parsing di documenti XML si può far riferimento alla pagina http://java.sun.com/webservices/docs/1.6/tutorial/doc/SJSXP2.html.

Gestire la mappa sul client

IL client (localizzaClima.jsp) deve far sì che ogni clic dell'utente venga intercettato. Inoltre si deve reperire le coordinate sulla mappa affinché queste vengano passate al server, il quale restituisce il tempo atmosferico per il punto di interesse. Per intercettare le coordinare è necessario creare un opportuno gestore; lo si può fare sul metodo load() caricato inizialmente:

```
function load() {
  if (GBrowserIsCompatible()) {
     map = new GMap2(document.getElementById("map"));
  map.addControl(new GSmallMapControl() );
  map.setCenter(new GLatLng(39.894,-95.888), 4); // USA
  GEvent.addListener(map, "click", function(overlay, latlng){
    chiamaServer(latlng);
  });
  }
}
```

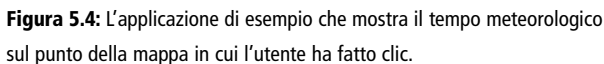
L'istruzione evidenziata fa sì che al click dell'utente sulla mappa, venga invocata la funzione (anonima) dichiarata come terzo parametro; tale funzione non fa altro che invocare chiamaServer a cui viene passato il punto su cui l'utente ha fatto click. Questa funzione

potrebbe essere scritta come:

```
function chiamaServer(latlng){
   mylat = latlng;
   GDownloadUrl(
    'proxy.jsp?lat='+latlng.lat()+'&lon='+latlng.lng(),
    function(data){
       var marker = creaMarker(mylat , data);
       map.addOverlay(marker);
       marker.openInfoWindowHtml(data);
    });
       return false;
}
```

La funzione non fa altro che invocare (via AJAX) proxy.jsp a cui passa le coordinate; al termine del caricamento del risultato viene invocata la funzione (anonima) usata come secondo parametro; tale funzione riceve come parametro (in questo caso chiamato "data") il contenuto reperito dall'invocazione AJAX; nell'esempio la funzione non fa altro che aprire un marker con i dati reperiti dal server. Tali dati (e lo si capisce guardando al codice di proxy.jsp, disponibile nei sorgenti allegati al libro) non sono altro che il codice HTML che va a comporre il contenuto del fumetto, ovvero una tabella HTML dove ogni riga rappresenta un'informazione meteorologica. In Figura 5.4 un esempio di esecuzione.
Ovviamente i dati restituiti dal server possono essere in diversi formati; nell'esempio appena visto essi non sono altro che informazioni HTML da mostrare; altre volte potrebbero essere documenti XML (è il caso in cui il client potrebbe desiderare di farci il parsing per reperire alcune delle tante informazioni restituite), altre volte potrebbero essere oggetti e/o array JavaScript, restituiti secondo l'usuale formato JSON. Un esempio verrà mostrato nella prossima applicazione…

Figura 5.4: L'applicazione di esempio che mostra il tempo meteorologico sul punto della mappa in cui l'utente ha fatto clic.

CALENDARI E PERCORSI, TUTTO CON GOOGLE!

Google, tra le numerose applicazioni fornite, permette di tener traccia dei propri appuntamenti usando una rubrica in rete. Tale applicazione si chiama Google Calendar ed è, come molte altre, completamente gratuita e accessibile in seguito ad una registrazione (si veda *http://www.google.com/calendar/*).

Inserire gli eventi

All'interno del proprio calendario (che, tra le altre cose, può essere visualizzato secondo molteplici viste: da quella giornaliera, a settimanale o mensile) è possibile inserire degli eventi. Un evento si caratterizza per titolo, giorno (compreso l'orario), la possibilità che sia ciclico (ovvero che si ripete nel tempo), una località e una descrizione; il tutto è gestito con una semplice interfaccia Web che si basa su AJAX. In particolare, per inserire un evento, è pos-

sibile fare clic su un punto qualsiasi del calendario e si apre un fumetto in cui l'intervallo temporale è quello dove si è fatto clic più 1 ora; inoltre viene chie

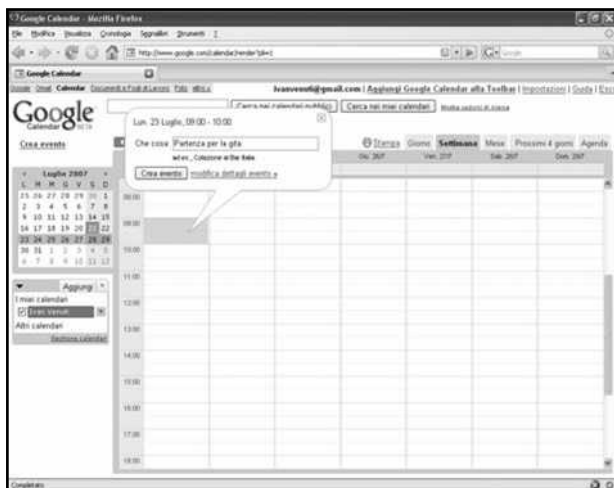



Figura 5.5: Creazione di un nuovo evento sul calendario con l'interfaccia semplificata.

Se, invece, si desidera inserire tutti i dettagli dell'evento, è necessario fare clic sul link "Modifica dettagli evento" o, in alternativa, fare clic su un evento già inserito; in entrambi i casi è possibile accedere a tutti i dettagli e valorizzarli (Figura 5.6). Per i nostri esempi è importante che il luogo dell'evento (campo "dove") possieda un indirizzo valido. Un evento può essere eliminato editandolo e poi facendo clic sul pulsante "Elimina" (pulsantiera in alto).
In Figura 5.7 tutti i dettagli di una ipotetica visita della Toscana di cinque giorni, che tocca Pisa, Lari, San Gimignano, Monteriggioni, Siena, Firenze, Pistoia, Lucca e Viareggio…

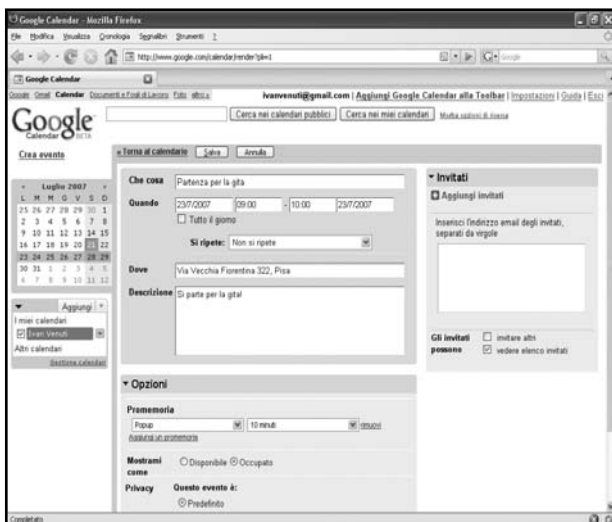



Figura 5.6: Dettagli di un evento.

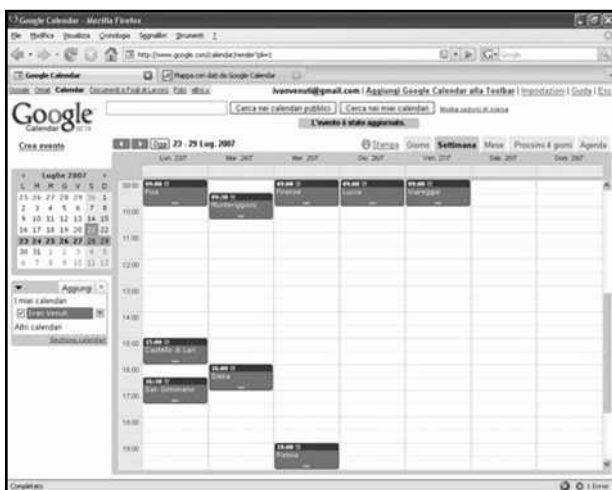



Figura 5.7: Una ipotetica gita in Toscana di cinque giorni.

Accedere agli eventi via API

Per accedere agli eventi del proprio calendario è necessario, per prima cosa, installare le librerie di accesso specifiche per Google Calendar; anch'esse sono contenuto nelle librerie GData, accessibili dalla pagina http://code.google.com/apis/gdata). Per l'esempio che si vuol creare è necessario permettere l'accesso alle località (indirizzi) memorizzate per gli eventi in un arco temporale (dalle ore 00:00 di un primo giorno alle ore 23:59 del secondo giorno specificato; facendo coincidere le due date si ottengono gli appuntamenti di una specifica giornata). La classe che implementa questa funzionalità è it.ioprogrammo.mashup.gdata.Calendar e, in particolare il metodo prendiEventi, che restituisce un oggetto di tipo EventFeed. Il metodo, dopo essersi autenticato al servizio, imposta una query che permette di restituire tutti gli eventi nell'arco temporale desiderato:

```
public static EventFeed prendiEventi(Date daData, Date aData){
  GoogleService myService = accediServizio();
  URL feedUrl = new URL(
     "http://www.google.com/calendar/feeds/"+
     gestChiavi.getValue("google-email")+"/private/full");
  CalendarQuery myQuery = new CalendarQuery(feedUrl);
  String str_inizio = DateManager.getDateJDBC(daData);
  String str_fine = DateManager.getDateJDBC(aData);
  myQuery.setMinimumStartTime(
     DateTime.parseDateTime(str_inizio+"T00:00:00"));
  myQuery.setMaximumStartTime(
     DateTime.parseDateTime(str_fine+"T23:59:59"));
  myQuery.addCustomParameter(
   new Query.CustomParameter( "orderby", "starttime"));
  myQuery.addCustomParameter(
   new Query.CustomParameter( "sortorder", "ascending"));
  myQuery.addCustomParameter(
   new Query.CustomParameter( "singleevents", "false"));
```

```
  new EventFeed().declareExtensions(
    myService.getExtensionProfile());
  EventFeed calFeed = (EventFeed)
    myService.query(myQuery, EventFeed.class);
  return calFeed;
}
```

Pubblicare gli indirizzi via JSON

La classe server appena illustrata deve essere acceduta dal Web. La pagina indirizzi-georeferenziati.jsp legge i parametri dalla request (data inizio e di fine), usa la classe precedente per reperire tutti gli eventi e, quello che deve fare, è reperire le località memorizzate in ciascun evento e inviarle alla pagina che userà gli indirizzi per disegnare una mappa; per comodità si invia la lista degli indirizzi usando la notazione JSON per la costruzione degli array, ovvero:

```
[ 'Elemento1', 'Elemento 2', …, 'Elemento N']
```

Ma andiamo con ordine; la JSP deve dapprima leggere i parametri della request e, se questi sono nulli o vuoti, impostarli con la data del sistema:

```
<%
  String dataDaStr = request.getParameter("dataDa");
  String dataAStr = request.getParameter("dataA");
  java.util.Date dataDa;
  java.util.Date dataA;
  if (dataDaStr!=null && dataDaStr.length()>0)
    dataDa = DateManager.getDateFromIta( dataDaStr );
  else
    dataDa = new java.util.Date();
  if (dataAStr!=null && dataAStr.length()>0)
    dataA = DateManager.getDateFromIta( dataAStr );
```

```
else
  dataA = new java.util.Date();
%>
```

Usando le due date prende gli eventi dal server:

```
<%
 EventFeed ris = Calendar.prendiEventi(dataDa, dataA);
%>
```

Non resta che scorrerli e comporre, su una variabile temporanea chiamata wString, il risultato e, in particolare, gli oggetti di tipo EventEntry acceduti usando ris.getEntries():

```
<%
 String wString = "";
 String add = "";
 if (ris.getEntries().size() > 0){
   java.util.List lLocs = null;
   EventEntry calEntry = null;
   for (int n = 0; n < ris.getEntries().size(); n++) {
     calEntry = (EventEntry) ris.getEntries().get(n);
     // accede alle località
   }
 }
%>
```

La parte precedentemente indicata con un commento, è la seguente, e si occupa di reperire le località (getLocations) e, iterando su di esse, accedere agli oggetti di tipo Where da cui ricavare il testo:

```
lLocs = calEntry.getLocations();
for (Iterator iterator1 = lLocs.iterator(); iterator1.hasNext();) {
```

```
  Where where = (Where) iterator1.next();
  wString = wString + add + "'"+where.getValueString()+"'";
  add = ", ";
}
```

Infine la JSP fa il seguente output:

```
[<%= wString %>]
```

Ovvero mette in formato JSON gli indirizzi reperiti (o stringa vuota se non ce ne sono). Di seguito la pagina che legge il risultato e lo mostra su una mappa, dopo aver georeferenziato gli indirizzi recuperati…

Mappa e georeferenziazione

La pagina che conclude l'applicazione è calendarioSuMappaConDirezioni.jsp. Tale pagina dovrà sicuramente contenere una form con due campi di inserimento testo: uno per ogni data (inizio e fine del periodo su cui interrogare il calendario); inoltre ci sarà un pulsante per "comandare" l'aggiornamento della mappa:

```
<form action="#">
 <input type="text" name="dataDa"
  value="<%= DateManager.getDateIta(new java.util.Date()) %>" />
 <input type="text" name="dataA"
  value="<%= DateManager.getDateIta(new java.util.Date()) %>" />
 <input type="button" name="percorso" value="Percorso"
  onclick="reloadAll()"/>
</form>
```

Si noti che alla pressione del pulsante viene invocate la funzione JavaScript reloadAll(); ecco cosa fa:

```
function reloadAll(){
```

```
var invoca =
  "indirizzi-georeferenziati.jsp?dataDa="+
  document.forms[0].dataDa.value+
  "&dataA="+document.forms[0].dataA.value;
GDownloadUrl(invoca, function(doc) {
  direzioni.loadFromWaypoints(eval(doc));
});
}
```

In pratica viene costruita un'URL che passa alla pagina indirizzi-georeferenziati.jsp i due valori inseriti nei campi di inserimento e tale URL viene invocata via AJAX. Il risultato dell'invocazione esegue il metodo loadFromWaypoints sull'oggetto direzione. Il metodo si preoccupa, attraverso una ulteriore chiamata AJAX ai server di Google, di georeferenziare la lista di indirizzi con cui viene invocato. L'oggetto "direzioni" deve essere così costruito (e inizializzato):

```
var direzioni;

function load() {
  if (GBrowserIsCompatible()) {
    // Altre inizializzazioni
    direzioni = new GDirections(map);
    reloadAll();
  }
}
```

Ovviamente map è, come sempre, una mappa costruita con Google Maps. Si noti anche che il metodo reloadAll viene invocato subito (in questo modo all'apertura della pagina vengono mostrate eventuali località presenti negli eventi del giorno corrente). In Figura 5.8 un esempio di esecuzione della pagina.
Si noti che per ogni tappa, sulla mappa viene evidenziato un marca-

tore. Facendovi clic appare il dettaglio dell'indirizzo specificato.

Figura 5.8: L'applicazione visualizza tutte le località specificate in un determinato giorno.

La possibilità di visualizzare le località specificate in eventi compresi tra due date potrebbe essere usata per visualizzare anche eventi futuri, come può essere l'organizzazione di una vacanza! In Figura 5.9, per esempio, una gita in Toscana organizzata usando Google Calendar e il cui itinerario è mostrato dall'applicazione appena costruita.

Attenzione

L'applicazione assume che gli indirizzi immessi siano non ambigui. Un caso di ambiguità è usare solo "Siena" come descrizione del luogo. Per verificarlo, dai dettagli dell'evento, seguire il link "Mappa". Google prova a disegnare la mappa corrispondente al luogo e, nel caso di ambiguità (Figura 5.10), specifica, sulla sinistra, le località che potrebbero essere quelle cercate ma che non riesce a distinguerle in automatico. Per evitare si può sia inserire una via (si provi con "Piazza del Campo") o lo stato ("Italia).

Figura 5.9: Un'ipotetica gita in Toscana mostrata dall'applicazione.

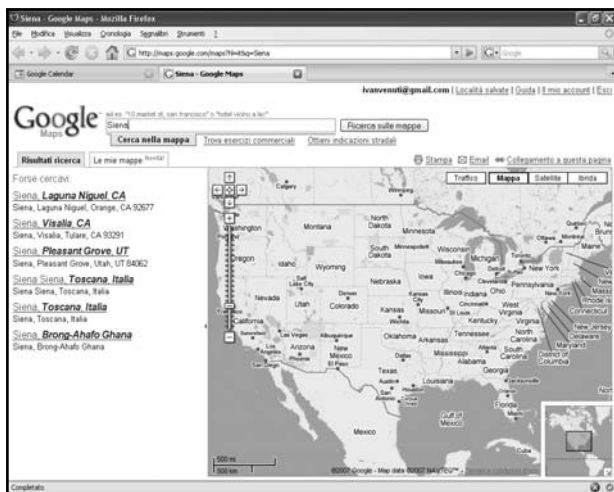



Figura 5.10: "Siena" risulta un indirizzo ambiguo.

Tempi di risposta

L'applicazione appena realizzata mette in evidenza possibili lentezze nel visualizzare le mappe. Il motivo è presto detto: prima di visualizzare la mappa (che comunque comporta una chiamata AJAX al server di Google!) il server deve reperire i dati da Google Calendar, pertanto deve attendere la risposta del server remoto. I dati vengono comunicati al client il quale, a sua volta, usa la georeferenziazione di Google usando, ancora una volta, un'invocazione AJAX al server di Google. Tutto questo porta a tempi non certo brevi e, cosa ancor più importante, in presenza di numerosi visitatori può rivelarsi un appesantimento del sistema e del consumo di risorse. Per evitarlo si potrebbe fare delle operazioni di cache locali. In particolare il server potrebbe, di tanto in tanto, interrogare Google Calendar e aggiornare la propria copia locale. In tale circostanza potrebbe ricorrere alla georeferenziazione (pertanto una volta per tutte, in quanto anche tale risultato potrebbe essere messo in cache!). Il client, a questo punto, potrebbe limitarsi ad un'unica interazione con il server locale.
Per applicazioni in ambienti di produzione questo tipo di valutazioni è di fondamentale importanza e vanno fatte prima ancora di rendere pubblico il servizio.

Aiuto sulle date?

L'esempio appena illustrato può essere migliorato anche facendo sì che le date inserite debbano essere valide. Un modo è verificarle dopo l'immissione. Altrimenti è possibile evitare l'inserimento diretto e, grazie a finestre popup html o a contenuto DHTML, proporre un calendario da dove, attraverso una selezione, immettere le date. Oltre a risolvere il problema della correttezza, anche l'interfaccia diventa più semplice e intuitiva. Lo sviluppo di questi componenti può, come sempre, essere fatta partendo da zero oppure adottare uno o più componeti (ed eventualmente personalizzarlo). Per esempio si veda le pagine http://www.dynarch.com/projects/calendar/ e *http://www.ja-*

vascriptkit.com/script/script2/epoch/index.shtml.

CONCLUSIONI

Fino ad ora sono state realizzate applicazioni che accedono a dati resi accessibili in maniera controllata e il loro “consumo”, anche grazie a librerie standard, è piuttosto semplice (o comunque semplificato).
Diverso il caso in cui i dati sono disponibili sul Web senza una precisa volontà di loro fruizione da parte di un agente automatico esterno. Nel prossimo capitolo verranno analizzati i possibili problemi e alcune soluzioni per aggirarli.

ACCESSO DIRETTO A RISORSE WEB

Come si è avuto modo di accennare in precedenza, quasi tutti i siti che vogliono condividere le informazioni pubblicate offrono una modalità di accesso ai dati orientata al contenuto. Talvolta così non è. Per esempio ci sono dei siti per cui è possibile utilizzare le informazioni reperite, ma che non offrono alcun modo vantaggioso per farlo. In questi casi l'unica possibilità è accedere alla pagina HTML dove compaiono i dati ed estrarre le informazioni. Tale estrazione può essere fatta con le usuali tecniche generali (accesso alla url e lettura "personalizzata" del contenuto) o grazie ad apposite librerie che sono state pensate per agevolare l'operazione di estrazione.

Attenzione
È necessario prestare attenzione ad eventuali vincoli sull'uso delle informazioni pubblicate, ancor più quando non sono fornite via API. Un esempio su tutti: si potrebbe pensare di usare il sito www.paginebianche.it per il reperimento (automatico) delle informazioni sui numeri di telefono dei nominativi reperiti da altre parti per creare un nuovo mashup. Questo uso non è consentito, in quanto esplicitamente vietato dalla licenza d'uso del sito, consultabile alla pagina http://www.paginebianche.it/execute.cgi?ts=16&tl=2&cb=&ep=copyright/tutelacopyright&es=&om=0

LIBRERIE JAVA: HTMLPARSER

Accedendo alla pagina *http://java-source.net/open-source/html-parsers* si ha a disposizione un elenco di alcune librerie (Open Source) per l'estrazione (e talvolta anche per la trasformazione) dei dati da pagine Web. Per estrazione si intende il reperimento delle informazioni contenute; per trasformazione si intende una qualche operazio-

ne che presenta le informazioni originali ma con struttura e/o presentazione diversa. Un caso particolare è la trasformazione di documenti HTML non ben formati in documenti ben formati (o addirittura XHTML valido!). Per i nostri scopi si ignorerà il problema della trasformazione, concentrandoci sul reperimento ed estrazione delle informazioni. In particolare si analizza la libreria "htmlparser".

Download e licenza d'uso

La pagina principale del progetto è raggiungibile all'indirizzo http://htmlparser.sourceforge.net. Il progetto è oltremodo interessante sia per la qualità del frame work proposto che per i notevoli (e semplici) strumenti grafici a corredo.
Attualmente la versione disponibile è la 1.6, il cui archivio compresso (comprensivo di sorgenti, tool di sviluppo e librerie) è di circa 4 Mb.
La licenza d'uso è la LGPL (pertanto è possibile usare la libreria anche all'interno di prodotti commerciali senza l'obbligo di rendere Open Source anch'essi!).
Una volta scompattato il file compresso si ha a disposizione una struttura composta da diverse cartelle; le principali sono:

docs/ : è la prima cartella da usare. Contiene la documentazione essenziale per partire (si inizia dall'usuale index.html!);
bin/ : contiene alcuni esempi e, tra gli altri, il tool filterbuilder, utilissimo per provare i filtri, creandoli in maniera grafica, per poi far loro generare i sorgenti della classe Java che li realizza;
lib/ : le librerie del progetto (da includere nel proprio classpath);

Inoltre è presente il file src.zip, che contiene tutti i file del progetto in formato sorgente (nonché la relativa documentazione in formato JavaDoc).

Usare filterbuilder

Un modo particolarmente semplice per prendere confidenza con le

potenzialità del frame work è quello di invocare lo script bin/filter-builder (usare quello con estensione .cmd se si utilizza un sistema Windows). La sua esecuzione fa sì che venga visualizzata una finestra grafica da cui è possibile realizzare (e provare) un qualsiasi filtro per l'estrazione di informazioni da una pagina Web. Come si può osservare dalla Figura 5.1, il tool si compone di due colonne; sulla prima (colonna di sinistra) troverà posto la rappresentazione grafica del filtro realizzato; sulla seconda (a destra) sarà possibile verificare l'esecuzione del filtro (l'esecuzione si riferisce ad una specifica pagina Web, il cui indirizzo è quello specificato sulla casella di testo, evidenziata in figura, in fondo alla finestra). È anche possibile navigare la struttura della pagina usando il comando "Operation > Fetch Page" (in Figura 6.1 è mostrato il contenuto della pagina http://www.inea.it/ssa/indfaceco.html, contenente gli indirizzi delle facoltà di Economia e commercio).

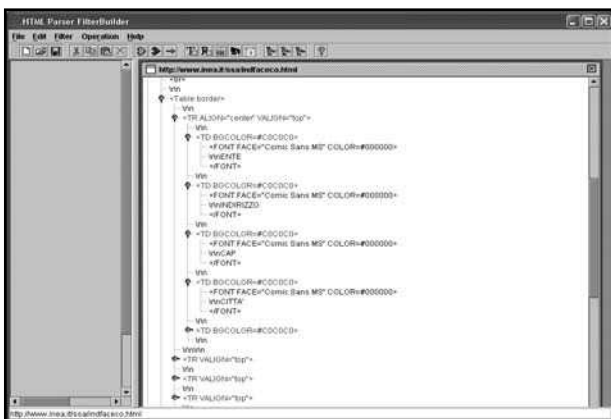



Figura 6.1: la finestra iniziale del tool filterbuilder con il contenuto di una pagina HTML.

Prima di partire è necessario analizzare la pagina HTML da cui si voglio-

no estrarre i dati e ricercare delle modalità di estrazione automatica. Il framewrok permette di specificare diversi tipi di filtri: si va da filtri che permettono pattern matching dei valori, a quelli che estraggono solo certi tipi di tag (per esempio TD per una casella all'interno di una tabella) con possibilità di testare la presenza di attributi particolari (come può esserlo width o class) aventi anche valori specifici (per esempio 100% su attributo width). Se si pensa ai documenti HTML come ad una gerarchia di tag (per esempio si pensi al tag <html> come il padre di tutti gli altri elementi, i tag <header> e <body> come a due possibili figli e così via), allora si comprende la possibilità offerta dal frame work di applicare uno dei filtri base o al nodo corrente o ad un suo predecessore o successore. Non solo: i diversi filtri possono essere composti con le usuali operazioni logiche (AND; OR e NOT). Per illustrare l'uso di un filtro si pensi al caso in cui si vogliano estrarre i valori delle facoltà di Economia e commercio dalla pagina http://www.inea.it/ssa/indfaceco.html (e, successivamente, dalla pagina http://www.inea.it/ssa/indfacagr.html che ha la stessa struttura ma con i valori per le facoltà di Agraria). L'analisi delle pagine Web evidenzia come i valori sono contenuti in una tabella e, precisamente, in tag <td>. Quello che si vuol fare è applicare un opportuno pattern matching ai contenuti il cui padre è proprio <td>. Ecco come si può procedere:

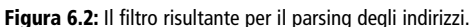
1) iniziare inserendo un filtro AND (menu "*Filter > AndFilter*");
2) selezionato il filtro AND applicare "*Filter > HashparentFilter*" affinché si possa specificare come deve essere fatto il padre;
3) inserire un filtro sul tag di tipo TD ("*Filter > TagnameFilter*") all'interno di HashparentFilter;
4) per l'altro predicando del filtro AND inserire un RegexFilter il cui valore è ".*"

Eseguendo "*Operation > Execute filter*" appaiono, per le pagine indicate, i contenuti estratti. Ora si vuol rendere persistente questo filtro al fine di poterlo utilizzare nelle proprie applicazioni. Scegliere "*File > Save*" e specificare il nome e il perscorso dove salvare il filtro (per

esempio dare il nome FiltraIndirizzi.java). In Figura 6.2 il filtro risultante, mentre di seguito il codice Java esportato:

```
public class FiltraIndirizzi
{
 public static void main (String args[])
 {
  TagNameFilter filter0 = new TagNameFilter ();
  filter0.setName ("TD");
  HasParentFilter filter1 = new HasParentFilter ();
  filter1.setRecursive (false);
  filter1.setParentFilter (filter0);
  RegexFilter filter2 = new RegexFilter ();
  filter2.setStrategy (RegexFilter.FIND);
  filter2.setPattern (".*");
  NodeFilter[] array0 = new NodeFilter[2];
  array0[0] = filter1;
  array0[1] = filter2;
  AndFilter filter3 = new AndFilter ();
  filter3.setPredicates (array0);
  NodeFilter[] array1 = new NodeFilter[1];
  array1[0] = filter3;
  FilterBean bean = new FilterBean ();
  bean.setFilters (array1);
  if (0 != args.length)
  {
   bean.setURL (args[0]);
   System.out.println (bean.getNodes ().toHtml ());
  }
  else
   System.out.println (
 "Usage: java -classpath .:htmlparser.jar FiltraIndirizzi <url>");
 }
```

```
}
```

Figura 6.2: Il filtro risultante per il parsing degli indirizzi.

Tale codice può essere anche modificato e adattato per ulteriori e future esigenze.
Sulla falsariga di quanto illustrato realizziamo un nuovo mashup…

MASHUP USANDO IL FORUM DI IOPROGRAMMO

Accedendo alla pagina http://forum.ioprogrammo.it/members.php?mode=view&boardid=0&by=userposts (Figura 6.3) è possibile conoscere gli utenti che hanno fatto più post all'interno dei forum della rivista IoProgrammo. Alcuni utenti hanno segnalato anche la propria pagina personale. Potrebbe essere interessante realizzare un mashup che legge la pagina, prende tutte le pagine Web degli utenti con più post e pubblichi una lista dove c'è uno screenshot della home page e un link verso di essa.

Figura 6.3: la pagina dove estrarre i dati.

Screenshot "dinamici"? WebThumb!

Un servizio che permette, dato un indirizzo di pagina Web, di averne uno screenshot è quello accessibile alla pagina http://bluga.net/webthumb/. Nella forma base (250 richieste di screenshot, o thumbnails, al mese) il servizio è gratuito, previa registrazione. La registrazione è pressoché immediata ed è importante copiare la chiave e farla accedere alla propria applicazione affinché si possano eseguire le richieste. Tali richieste possono essere fatte dal sito Web ma, cosa interessante per il nostro contesto, anche attraverso invocazioni di tipo REST. L'url da invocare è:
http://webthumb.bluga.net/api.php
A tale URL vanno inviati opportuni messaggi XML che fungono da comandi. Il comando base è così formato:

```
<webthumb>
 <apikey>key ottenuta dalla registrazione</apikey>
 <!—
  qui vanno altri parametric
```

```
  specifici per i diversi comandi
 -->
</webthumb>
```

I parametri specifici (che vanno inseriti al posto del commento del comando base) possono essere quelli elencati nel seguito. Si vedranno anche i dettagli della classe Java it.ioprogrammo.mashup.GestoreWebThumb; essa implementa l'interfaccia verso il servizio.

Nuova thumbnail

Ecco come deve essere composto il comando "request", usato per richiedere la generazione di una nuova thumbnail

```
<request>
 <url>ivenuti.altervista.org</url>
</request>
```

Il comando può avere (opzionalmente) specificate le dimensioni con cui viene aperta la pagina (in pratica è la dimensione della finestra del browser da cui fare lo screenshot):

```
<request>
 <url>ivenuti.altervista.org</url>
 <width>800</width>
 <height>600</height>
</request>
```

All'interno di una singola chiamata possono essere elencate più richieste. Per semplicità le classi Java che realizzano le richieste accettano una sola URL per chiamata; ecco il metodo che la gestisce:

```
public String nuovaRichiesta(String weburl){
  weburl = normalize(weburl);
```

```
String tmp;
if ((tmp=p.getProperty(weburl))!=null)
  return tmp;
String str = "<webthumb><apikey>"+apikey+"</apikey>"+
   "<request><url>"+weburl+"</url>"+
   "</request></webthumb>";
try {
  String key = parsingRisposta(
    sendCommand(str)
   );
  if (key!=null)
    p.setProperty(weburl, key);
} catch (Exception e) {
  e.printStackTrace();
}
return tmp;
}
```

Il "cuore" della funzione è rappresentato dal testo evidenziato: sendCommand è un metodo che, per comodità, realizza l'invocazione all'URL remota e si preoccupa di aprire lo stream per leggere la risposta:

```
public java.io.InputStream sendCommand(String command){
 try {
  URL url = new URL(urlService);
  URLConnection conn = url.openConnection();
  conn.setDoOutput(true);
  OutputStream os = conn.getOutputStream();
  os.write(command.getBytes());
  os.close();
  if (conn.getContentType().equals(CONTENT_TYPE_ERRORE)){
     this.dump(conn.getInputStream(),
       new BufferedOutputStream(System.err));
```

```
      return errorFile();
    }
    return conn.getInputStream();
  } catch (Exception e) {
    return errorFile();
  }
}
```

Si noti come il metodo verifica se c'è stato un errore in base al tipo di risposta (ovvero al suo content-type). In caso di errore viene comunque inviata un'immagine, ma si tratta di una immagine locale con una scritta di errore. Il metodo dump è un metodo (pubblico, affinché possa essere usato anche da classi esterne!) che esegue un semplice "travaso" di byte tra lo stream di output a quello di input.
La risposta ottenuta viene analizzata dal metodo parsingRisposta. Per eseguire il parsing è necessario comprendere come questa viene generata; una nuova richiesta, se soddisfatta, ottiene in risposta del codice XML al cui interno c'è un codice che d'ora in poi dovrà essere usato per identificare univocamente la thumbnail ed è contenuto come valore di un tag <job>:

```
<webthumb>
<jobs>
 <job estimate='20' time='2007-08-30 08:49:45'
                url='http://ivenuti.altervista.org'>wt4680b94abdf30</job>
</jobs>
</webthumb>
```

In questo caso la chiave da associare all'url è il testo evidenziato: wt4680b94abdf30.

Lo stato di una richiesta

Una limitazione, se così si può dire, del servizio è quella che la thumb-

nail non è disponibile subito, ma dopo un periodo di tempo; in alcuni casi potrebbe essere utile utilizzare la stima o procedere alla verifica dello stato inserendo, nel comando base, la seguente richiesta:

```
<status>
 <job>wt4680b94abdf30</job>
</status>
```

Nell'implementazione Java verrà ignorato lo stato; in presenza di errori sarà mostrata un'immagine di errore predefinita.

Reperire l'immagine

Infine ecco il metodo che, data la chiave, ne reperisce l'immagine (in un formato specificato come parametro); il comando inviato è evidenziato in grassetto:

```
public java.io.InputStream getKeyThumb(String key, String size){
 String cmd =
  "<webthumb>"+
   "<apikey>"+apikey+"</apikey>"+
   "<fetch>"+
     "<job>"+key+"</job>"+
     "<size>"+size+"</size>"+
   "</fetch>"+
  "</webthumb>";
 try {
   return sendCommand(cmd);
 } catch (Exception e) {
   return errorFile();
 }
}
```

Recuperare i siti Web degli utenti

L'esempio che si vuol realizzare implica che si esegua il parsing della pagina e, ottenuti gli indirizzi delle pagine personali, si usi la classe appena creata per la generazione delle thumbnail. Il parsing verrà fatto con HTML Parser (in particolare con htmlbuilder, come mostrato in Figura 6.4).

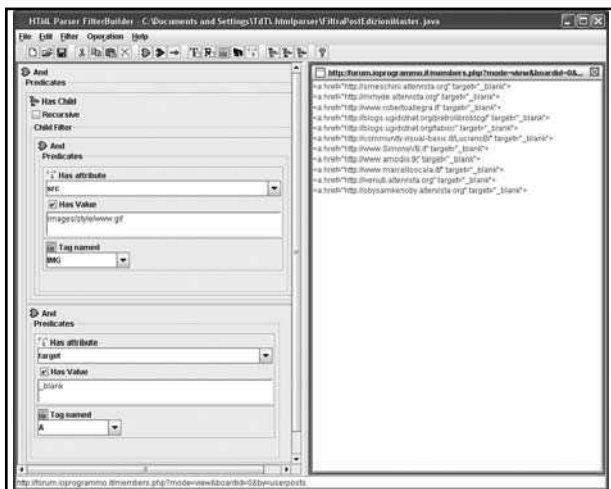



Figura 6.4: Il filtro per reperire le home page degli utenti.

Il codice Java, dopo essere stato esportato, può essere personalizzato; ecco il metodo re ingegnerizzato risultante (classe it.ioprogrammo.mashup.htmlparser.FiltraPostEdizioniMaster):

```
public static String[] getUrlMaggioriPost(String url){
 if (url==null)
   url = defaultUrl;
 HasAttributeFilter filter0 = new HasAttributeFilter ();
 filter0.setAttributeName ("src");
 filter0.setAttributeValue ("images/style/www.gif");
```

```
TagNameFilter filter1 = new TagNameFilter ();
filter1.setName ("IMG");
NodeFilter[] array0 = new NodeFilter[2];
array0[0] = filter0;
array0[1] = filter1;
AndFilter filter2 = new AndFilter ();
filter2.setPredicates (array0);
HasChildFilter filter3 = new HasChildFilter ();
filter3.setRecursive (false);
filter3.setChildFilter (filter2);
HasAttributeFilter filter4 = new HasAttributeFilter ();
filter4.setAttributeName ("target");
filter4.setAttributeValue ("_blank");
TagNameFilter filter5 = new TagNameFilter ();
filter5.setName ("A");
NodeFilter[] array1 = new NodeFilter[2];
array1[0] = filter4;
array1[1] = filter5;
AndFilter filter6 = new AndFilter ();
filter6.setPredicates (array1);
NodeFilter[] array2 = new NodeFilter[2];
array2[0] = filter3;
array2[1] = filter6;
AndFilter filter7 = new AndFilter ();
filter7.setPredicates (array2);
NodeFilter[] array3 = new NodeFilter[1];
array3[0] = filter7;
FilterBean bean = new FilterBean ();
bean.setFilters (array3);
bean.setURL (url);
String[] ris = new String[bean.getNodes().size()];
for(int i=0; i<bean.getNodes().size(); i++){
  String tmp = bean.getNodes().elementAt(i).getText();
```

```
   int primo = tmp.indexOf("\" ");
   ris[i] = tmp.substring(primo+1,
     primo+tmp.substring(primo+1).indexOf("\" ")+1);
  }
  return ris;
}
```

Mostrare le immagini

Un'immagine, una volta reperita dal server remoto, deve essere mostrata al client. Un modo è quella di salvarla in locale; altro modo (utile però solo per questo esempio, perché, come si è già avuto modo di dire, è sempre meglio eseguire il cache dei risultati!) è quello di passare direttamente lo stream di output del server remoto al client che ha fatto la richiesta. Ecco la servlet che realizza una simile funzionalità:

```
public class GetImageServlet extends HttpServlet {
  public static final String KEY = "key";
  public static final String SIZE = "size";
  protected void doGet(
   HttpServletRequest request, HttpServletResponse response){
    String src = request.getParameter(KEY);
    String size = request.getParameter(SIZE);
    if (size == null){
      size = GestoreWebThumb.SIZE_SMALL;
    }
    GestoreWebThumb thumb = GestoreWebThumb.getInstance();
    response.setContentType("image/jpeg");
    try {
      java.io.InputStream is
       = thumb.getKeyThumb(src, size);
      thumb.dump(is, response.getOutputStream() );
      is.close();
```

```
    response.flushBuffer();
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

Anche in questo casi si fa uso del metodo dump per l'invio dei dati. È importante osservare come la servlet specifichi correttamente il content type nella risposta http al client. Tra i parametri letti ci sono sia la chiave da cui ricavare l'immagine, sia il suo formato (se quest'ultimo parametro è assente, si esegue il download dell'immagine nel formato più piccolo).

La pagina di visualizzazione

Ecco come si potrebbe realizzare la JSP per la visualizzazione delle thumbnail. Per esempio si potrebbe mostrare il primo sito Web (quello che si riferisce all'utente con più post) in formato medio/grande:

```
<%
 GestoreWebThumb gestore = GestoreWebThumb.getInstance();
 String[] qualiUrl = FiltraPostEdizioniMaster.getUrlMaggioriPost(null);
%>
  <tr><td colspan="<%= numeroColonne %>">
    <a href="<%= qualiUrl[0] %>" target="_blank"
 ><img
   src="./servlet/image/download.jpg?size=<%= GestoreWebThumb.SIZE
 _MEDIUM_LARGE %>&key=<%= gestore.nuovaRichiesta( qualiUrl[0] ) %
                                                                 >"
   alt="<%= qualiUrl[0] %>"
   title="<%= qualiUrl[0] %>"
   <%= GestoreWebThumb.xy_SIZE_MEDIUM_LARGE %> /></a>
  </td></tr>
 <tr>
```

Si noti come la sorgente dell'immagine è una url a cui risponde la servlet descritta in precedenza. Non resta che stampare tutti gli altri siti Web reperiti (stavolta in formato medio suddivise per tre thumbnail per riga):

```
<%
 int numeroColonne = 3;
 int i=0;
 for(i=1; i<qualiUrl.length; i++){
%>
 <td>
 <a href="<%= qualiUrl[i] %>" target="_blank"
 ><img
   src="./servlet/image/download.jpg?size=<%= GestoreWebThumb.SIZE
        _MEDIUM %>&key=<%= gestore.nuovaRichiesta( qualiUrl[i] ) %>"
   alt="<%= qualiUrl[i] %>"
   title="<%= qualiUrl[i] %>"
   <%= GestoreWebThumb.xy_SIZE_MEDIUM %> /></a>
 </td>
<%
  if(i%numeroColonne==0){
%>
 </tr>
 <tr>
<%
  }
 }
 </tr>
 </table>
```

Migliorare il risultato

La prima invocazione alla JSP di visualizzazione può essere una pagina piena di errori (Figura 6.5). Questo perché la cattura delle thumbnail non è immediata ma necessita di un po' di tempo. -

Figura 6.5: Solo errori? Il server sta prendendo le thumbnail!

Una possibile (semplice) soluzione consiste nel richiedere la generazione delle thumbnail allo startup della webapp. Infatti è probabile che gli utenti con più post siano stabili nel breve periodo, rendendo rare eventuali nuove richieste (e in tal caso è accettabile un errore momentaneo). Ecco come si può realizzare una servlet che esegue tale inizializzazione:

```
public class StartupServlet extends HttpServlet {
 public void init(ServletConfig config) throws ServletException {
   GestoreWebThumb.getInstance().setApikey( chiave);
   FiltraPostEdizioniMaster.loadAndStoreNewMembers();
 }
 // altro
}
```

è importante che il file /WEB-INF/web.xml contenga l'attributo load-on-startup per tale servlet:

```
<servlet>
```

```
<servlet-name>initServlet</servlet-name>
<servlet-class>
  it.ioprogrammo.mashup.servlet.StartupServlet
</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
```

Che accade per le nuove immagini generate durante l'esecuzione? Queste vengono memorizzate in una struttura in memoria; per renderle persistenti vanno salvate. Un modo è quello di salvare la struttura ogni volta che questa viene aggiornata. Altrimenti si potrebbe implementare il metodo destroy() che viene invocato quando la webapp viene fermata (sia perché viene fermato il Servlet Container, sia per eventuali nuovi deploy o riavvie del contesto):

```
public void destroy(){
  GestoreWebThumb.getInstance().storeKeys();
}
```

Il risultato complessivo? In Figura 6.6 fa bella mostra si sé!

Figura 6.6: L'applicazione finale

ALTRI MASHUP IN PHP

UN AGGREGATORE DI FEEDS RSS PER YOUTUBE

YouTube è essenzialmente un servizio che permette di mostrare i video. La sua enorme popolarità è dovuta al fatto che permette l'upload di video creati da chiunque. Ospita quindi materiale delle più disparate provenienze che non sia coperto da diritti. Anche YouTube, al pari di Google, ha sviluppato delle API tramite le quali è possibile accedere all'insieme dei suoi video.

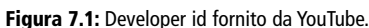
Dal sito YouTube: "*YouTube offre il completo accesso alle parti principali della sua raccolta di video e alla sua community mediante un'interfaccia API aperta e feed RSS. Con l'utilizzo delle nostre API, puoi facilmente integrare nella tua applicazione dei video online dalla raccolta di video di YouTube, che cresce ogni giorno di più. Una volta creato un profilo sviluppatore, sei pronto a sfruttare al massimo le potenzialità di YouTube.*"

L'esempio descritto in questo capitolo consiste in un tool che permette l'estrazione di feed dal sito YouTube, ne effettua il parsing e visualizza i contenuti ottenuti in base alle preferenze dell'utente. Tutta la logica viene racchiusa in una classe PHP di facile utilizzo.

Ottenere un developer ID

Anche YouTube richiede una chiave per utilizzare le sue API; tale chiave viene, in genere, chiamata developer id. Per ottenerla occorre creare un account su YouTube e, successivamente, effettuare la login e connettersi alla pagina http://www.youtube.com/my_profile_dev. Una volta compilata la form in figura 7.1 si ottiene il developer id (valido per il dominio immesso) che permette l'accesso alle API.

Figura 7.1: Developer id fornito da YouTube.

Cosa sono i feed RSS

Prima di iniziare lo sviluppo dell'esempio occorre descrivere brevemente il significato di feed RSS (Really Simple Syndication). In estrema sintesi possiamo dire che si tratta di un formato basato su XML il cui scopo è la diffusione di contenuti. Un utente, iscrivendosi ai feed RSS di un sito (meglio: di una fonte di contenuti online), ha la possibilità di ricevere in automatico informazioni aggiornate e personalizzate senza dover effettuare periodici controlli del sito in questione per ottenere eventuali aggiornamenti. L'applicazione descritta nel presente capitolo si occupa di estrarre documenti RSS dal sito YouTube, di effettuarne il parsing (individuando e decodificando i diversi elementi del documento XML) per poi convertire i contenuti decodificati in formato HTML. In questo modo sarà possibile incorporare i contenuti desiderati di YouTube all'interno di un qualsiasi sito.

Feed RSS di YouTube: per tag e per utenti

YouTube offre diversi feed RSS, ognuno per gruppi di video suddivisi per categorie: i video più visti, i video caricati di recente, i video il cui titolo contiene una certa parola, ecc...I feed sono personalizzati

per utenti e per tag.La richiesta di una lista di video associati ad un utente ha il seguente formato:

http://www.youtube.com/api2_rest?method=youtube.videos.list_by_user

Cui vanno appesi i parametri dev_id e user, rispettivamente la chiave fornita da YouTube e l'utente associato ai video scaricabili.
Mentre per ottenere una lista di video associati ad un tag occorre comporre la stringa che segue:

http://www.youtube.com/api2_rest?method=youtube.videos.list_by_tag

I parametri sono, in questo caso, dev_id e tag, che identificano la chiave fornita da youtube ed il tag per cui si desidera ottenere la lista.

Per esempio, la risposta fornita da youtube per la richiesta *http://www.youtube.com/api2_rest?method=youtube.videos.list_by_tag&dev_id=unqOBlxyzjA&tag=paintball* è un documento in formato XML del tipo:

```
<?xml version="1.0" encoding="utf-8" ?>
<ut_response status="ok">
<video_list>
  <total>23187</total>
<video>
  <author>cykovisuals</author>
  <id>ddMowxKchko</id>
  <title>Paintball Headshot</title>
  <length_seconds>33</length_seconds>
  <rating_avg>4.81</rating_avg>
  <rating_count>3185</rating_count>
  <description>Some uncovered kid gets owned by a paintball to the dome.
                                        cykovisuals dot com</description>
```

```
  <view_count>1097250</view_count>
  <upload_time>1165037719</upload_time>
  <comment_count>3135</comment_count>
  <tags>headshot head shot paintball paint ball airgun tippman angel
                                        spray cykovisuals pbaz</tags>
  <url>http://www.youtube.com/?v=ddMowxKchko</url>
 <thumbnail_url>http://img.youtube.com/vi/ddMowxKchko/default.jpg
                                                  </thumbnail_url>
  </video>
...
</video_list>
 </ut_response>
```

Occorre quindi, ancora una volta, effettuare una richiesta HTTP e parserizzare la risposta strutturata per ottenere le informazioni desiderate. Anche in questo caso ci serviremo del pacchetto CURL per effettuare la richiesta a YouTube e della libreria SimpleXML per la convalida e l'estrazione dei dati dalla risposta fornitaci.

UNA FUNZIONE PER LA LETTURA DI FEEDS.

Alla base del nostro lettore di feed vi è la funzione getFeed che, sempre mediante l'utilizzo del pacchetto CURL, introdotto nel Capitolo 2, effettua la richiesta di un feed, in uno dei formati specificati in precedenza, e restituisce il documento XML di risposta.

```
function getFeed($feed){
  $curl_object = curl_init();
  $timeout = 0;
  curl_setopt ($object, CURLOPT_URL, $feed);
  curl_setopt ($object, CURLOPT_RETURNTRANSFER, 1);
  curl_setopt ($object, CURLOPT_CONNECTTIMEOUT, $timeout);
```

```
  $youtube_response = curl_exec($curl_object);
  curl_close($ch);
  return $youtube_response;
}
```

La funzione parseYTResp, invece, si occupa del parsing della risposta e della sua trasformazione in formato HTML, in modo da poter essere immediatamente inclusa in una pagina web.

```
function parseYTResp($youtube_response, $number_of_video){

  $yt_result = simplexml_load_string($youtube_response);

  /*
    Parsing della risposta XML fornita da YuoTube, il risultato è
    inserito negli array $url,$thumb_nail,$description, $video
  */

  // Effettua un encoding HTML di tutte le descrizioni
  for($i=0;$i<count($description[0]);$i++){

  $description[0][$i] = preg_replace("/&#60;/","<",$description[0][$i]);
  $description[0][$i] = html_entity_decode($description[0][$i],ENT_
                                                          QUOTES);
  }

  // Numero di video ritornati
  $total_videos = count($video[0]);
  if($number_of_video > $total_videos or $number_of_video == 0){
    $number_of_video = $total_videos;
}

  //Formattazione HTML
```

```
 $html_to display = “”;
 for($i = 0; $i<$howmany; $i++){

   $html_to display .= "<p><a href=\"".$url[0][$i]."\" target=\"_blank\"
><img src=\"".$thumb_nail[0][$i]."\"></a>".$description[0][$i]."</p>";
   }
   return $html_to_display;
}
```

Viene effettuato un parsing della risposta XML fornita da YouTube utilizzando la libreria SimpleXML. Il risultato è inserito negli array $url, $thumb_nail, $description, $video. Questi sono utilizzati per comporre la parte HTML da incorporare nella pagina web.
Dal frammento dell’esempio di risposta visto in precedenza la funzione restituisce il seguente codice HTML:

```
<p><a href=”http://www.youtube.com/?v=ddMowxKchko" target="
                                                    _blank">
<img src=”http://img.youtube.com/vi/ddMowxKchko/default.jpg"></a>
Some uncovered kid gets owned by a paintball to the dome. cykovisuals dot
                                                    com </p>
```

La classe YouTubeFeed contiene tre funzioni principali che utilizzano le due funzioni appena analizzate

```
getVideoByFeature($number_of_video);
getVideoByUser($user, $number_of_video);
getVideoByTag($tag, $number_of_video);
```

Ognuna di queste funzioni utilizza parametri differenti per estrarre una lista di video utilizzando le API YouTube. Ogni funzione necessita del developer_id fornito da YouTube, si ipotizza che sia salvato all’interno del file YouTubeDevID.txt situato nella stessa directory

del file YouTubeFeed.php contenente il codice in esame.
A titolo di esempio ne viene di seguito descritta una. Le altre differiscono solo per il metodo delle API YouTube indicato. Viene composta la richiesta per le API YouTube utilizzando il metodo videos.list_by_user; la richiesta è inviata tramite la funzione getFeed e la risposta viene passata alla funzione

```
function getVideoByUser ($user, $number_of_video){
  // Carica il developer id fornito da YouTube
$fh = fopen("YouTubeDevID.txt", "r");
if (!$fh) die("Errore: YT Developer ID non trovato");
$yt_dev_id = fread($fh, filesize("YouTubeDevID.txt"));
fclose($fh);
  // Compone la richiesta per le YuoTube API
  $feed = “http://www.youtube.com/api2_rest?”;
  $feed .= “method=youtube.videos.list_by_user”;
  $feed .= “&dev_id=$yt_dev_id&user=$user";
  $yt_response = $this->getFeed($feed);

  $html_code = $this->parseYTResp($yt_response, $number_of_video);
  return $html_code;
}
```

La pagina web che utilizza la classe contiene il seguente codice:

```
$you_tube = new YouTubeFeed;
echo($youtube-> getVideoByFeature(0));
```

Questo esempio mostra tutti i video estratti dal metodo youtube.videos.list_featured delle API di YouTube, mentre gli esempi:

```
echo($youtube->getVideoByTag(“paintball”, 0));
echo($youtube->getVideoByUser(“cykovisuals”, 0));
```

Mostrano, rispettivamente, tutti i video estratti dal metodobyoutube.videos.list_by_tag utilizzando come parametro il tag "paintball" e youtube.videos.list_by_user utilizzando come parametro l'user cykovisuals. Nella figura seguente (Figura 7.2) viene visualizzato il risultato delle operazioni descritte all'interno di una pagina HTML.

Figura 7.2: Come appare l'applicazione

LAVORARE CON FLICKR

Flickr è un servizio, raggiungibile all'URL *www.flickr.com* , che permette all'utente, dopo essersi registrato, di utilizzare le proprie foto per creare veri e propri cataloghi on-line. Flickr ha acquistato una notorietà sempre crescente tra gli utenti del web per la molteplicità e la potenza delle funzioni che offre e, soprattutto, per la facilità con cui è possibile accedere a tali funzioni. È possibile, infatti, inserire foto e catalogarle per argomento od autore, visualizzarle ed effettuarvi interventi di fotoritocco. Oltre a questo, Flickr mette a disposizione degli sviluppatori delle API che forniscono la possibilità di accedere alle foto e di utilizzarle sul proprio sito web
Argomento di questo capitolo saranno proprio queste interfacce; ne spiegheremo le modalità di utilizzo tramite due esempi che, utiliz-

zando funzionalità diverse, illustreranno come mostrare nel proprio sito gruppi di immagini provenienti da Flickr. Per poter utilizzare le API di Flickr - così come per Google, YouTube ed altri servizi del genere – è necessario procurarsi una Flickr Key, che va richiesta direttamente a Flickr. Vediamo di che cosa si tratta e come fare per procurarsela.

Ottenere una Flickr API Key

I servizi che Flickr mette a disposizione degli utenti sono gratuiti ma, come accennato poc'anzi, per essere abilitati all'utilizzo delle API, occorre essere provvisti di una chiave, comunemente chiamata Flickr API Key. Si tratta di un codice alfanumerico che identifica univocamente l'utilizzatore che andrà ad usufruire dei servizi di Flickr. Ogni qual volta viene effettuata una chiamata ai metodi di Flickr, questa deve contenenere, tra gli altri parametri, la chiave di riconoscimento. Per ottenere una Flickr API Key è necessario creare un account sul sito e, successivamente, accedere alla pagina http://www.flickr.com/services/api/keys/apply indicando l'utilizzo che si intende fare dei servizi che Flickr metterà a disposizione. Verrà generata una stringa di lettere e numeri piuttosto lunga, che andrà fornita ai servizi Flickr ogni volta che se ne effettuerà la chiamata. Insieme alla chiave viene fornito un identificativo segreto, necessario solamente nel caso si desideri inserire foto in Flickr. Andiamo adesso ad introdurre phpFlickr, uno strumento scritto in PHP, che utilizzeremo nei nostri esempi e che, come si vedrà, facilita significativamente l'utilizzo delle API Flickr.

PHPFlickr, una classe per accedere a Flickr

Analogamente a quanto visto per altre tipologie di interfacce per l'accesso a servizi web, quali Google Maps e YouTube, la risposta che viene fornita è strutturata in un particolare dialetto XML stabilito dal fornitore del servizio. Una volta effettuata la richiesta, quindi, lo sviluppatore deve effettuare il parsing della response XML ed estrarre da essa

le informazioni che intende utilizzare. È, questa, una metodologia spesso obbligata e che, più volte, è stata utilizzata in questo libro. Per utilizzare le API di Flickr si opterà invece per un approccio differente, sfruttando un pacchetto free, chiamato PhpFlickr. PhpFlicker è una classe, scritta da Dan Coulter e scaricabile dall'indirizzo *http://phpflickr.com*, che agisce da wrapper nei confronti delle API Flickr. Si occupa di comporre le richieste, inoltrarle a Flickr e di processare le risposte XML ottenute, ritornando i dati in un array manipolabile in maniera facile ed intuitiva.

Installazione di PHPFlickr

Per ottenere PhpFlicker è necessario connettersi al sito *http://phpflickr.com* ed effettuare il download del pacchetto. Attenzione: è preferibile scaricare una versione successiva alla 1.3.1 in modo da non dover installare anche la libreria PEAR (http://pear.php.net) che PhpFlickr utilizza per effettuare le connessioni HTTP. Una volta effettuato il download non resta che copiare i files sotto una directory della propria applicazione. La classe risulterà immediatamente disponibile.

Struttura ed utilizzo di PHPFlickr

Come detto inprecedenza PHPFlickr agisce da wrapper nei confronti delle API di Flickr, rendendole trasparenti all'utilizzatore ed occupandosi di tutta la logica di trasmissione/elaborazione delle informazioni. Tutti i metodi forniti dalle API di Flickr sono implementati in PhpFlickr. La lista completa dei metodi forniti dalle API Flickr è disponibile all'URL *http://www.flickr.com/services/api/*. Per invocare un metodo tramite PHPFlickr è sufficiente togliere la parola "flickr" e sostituire i punti con caratteri underscore. Così, per esempio, se si desiderasse invocare il metodo

```
flickr.photos.search
```

Occorrerebbe, una volta instanziata un oggetto phpFlicke, utilizzarne il metodo:

```
$php_flickr = new phpFlickr();
$php_flickr->photos_search()
```

Oppure per

```
flickr.photos.licenses.getInfo
```

Si dovrà utilizzare:

```
$php_flickr->photos_licenses_getInfo()
```

Occorre fare attenzione al fatto che la sintassi è case sensitive: le lettere scritte in maiuscolo devono rimanere tali, così come quelle in minuscolo. Dopo questa breve disamina su PhpFlickr passiamo ad un esempio pratico di utilizzo.

Un primo esempio

Il primo esempio di utilizzo delle API Flickr tramite PhpFlickr consiste nel chiedere un determinato numero di foto appartenenti ad un

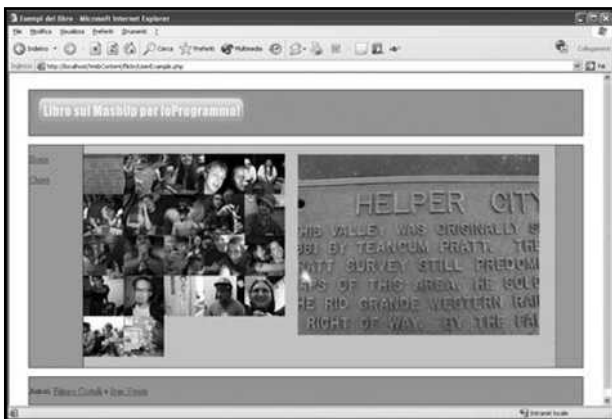



Figura 7.3: Come si presenta l'applicazione.

certo utente e visualizzare nella parte sinistra di una pagina web le icone relative alle foto ottenute. Selezionando con il mouse l'icona, la foto corrispondente viene visualizzata nella parte destra della pagina. L'aspetto della pagina è quello rappresentato in figura 7.3 Passiamo ad analizzare le parti significative del codice:

```
<?php
require_once("phpFlickr/phpFlickr-2.1.0/phpFlickr.php");

$fh = fopen("FlickrApiKey.txt", "r");
if (!$fh)
{
  die("Non si può trovare la chiave");
}
$api_key = fread($fh, filesize("FlickrApiKey.txt"));
fclose($fh);

$f = new phpFlickr($api_key);
$i = 0;
```

Il significato del primo frammento di codice è abbastanza chiaro: semplicemente viene letta la chiave di accesso alle API Flickr e viene instanziato un oggetto della classe PhpFlickr.

```
if (!empty($_POST['username'])) {
  $person = $f->people_findByUsername($_POST['username']);

  $photos_url = $f->urls_getUserPhotos($person['id']);

  $photos = $f->people_getPublicPhotos($person['id'],NULL, $_POST
                                                        [photos]);
```

Se il parametro 'username', trasmesso dalla form richiedente, non è

vuoto, allora viene utilizzato il metodo people_findByUsername (che va ad effettuare una chiamata, come detto, al metodo flickr.people.findByUsername) per ottenere l'identificativo che Flickr ha assegnato a tale utente. Ottenuto tale valore lo si utilizza per ottenere le foto pubbliche (nel numero specificato dalla form di input) inserite dall'username in oggetto. Questo viene fatto semplicemente utilizzando il metodo people_getPublicPhotos. Si noti come l'utilizzo di phpFlickr semplifichi notevolmente la vita, in quanto le informazioni relative alle foto richieste sono rese immediatamente disponibili in un array senza che il programmatore debba effettuare alcun lavoro di elaborazione sulla risposta del servizio. A questo punto non resta che scorrere l'array e generare l'output HTML adeguato.

```
$i = 0;
$count = 0;
$html = "";
$js = "";
 foreach ((array)$photos['photo'] as $photo) {
   $html .= '<a href="#" onClick ="';
   $html .= 'changePhoto('.$count].');">';
   $html.= '<img border="0" alt="'.$photo[title].'" ' . 'src=' .
   $f->buildPhotoURL($photo, "Square") . '>';
   $html.= '</a>';
   $js.= "photos[$count]=\"".$f->buildPhotoURL($photo, "Medium")."\";\n";
   $count++;
   $i++;
      if ($i % 6 == 0) {
         $html .= "<br>\n";
      }
   }// foreach
}
echo $_html;
?>
```

Viene scorso l'array delle foto e costruito l'html che dispone le foto nel formato di icone (utilizzando il metodo buildPhotoUrl con parametro "Square"). Selezionando un'icona viene richiamato il metodo Javascript changePhoto() con parametro il numero della foto. Tra un attimo vedremo in che cosa consiste tale metodo. Per ora evidenziamo che, come il lettore avrà probabilmente osservato, nel frammento di codice precedente viene costruita una stringa, assegnata alla variabile $js, che contiene un pezzo di codice Javascript. Tale stringa viene inserita nella funzione Javascript riportata qui sotto, il cui compito è quello di assegnare le foto ad un array Javascript. Questo array contiene l'url della foto e la sua formattazione come "Medium", come si vede dal metodo PhpFlickr utilizzato.

```
<script language="JavaScript1.1">
var photos=new Array();
var which=0;
<?php echo $js ?>
</script>
```

La pagina HTML contiene, inoltre, un tag IMG denominato phptoslider

```
<img src=" " name="photoslider">
```

A questo punto rimane da chiarire la funzione Javascript changePhoto, invocata quando viene selezionata un icona. Essa, semplicemente, assegna come source all'immagine 'photoslider' la foto corrispondente all'url contenuto nell'array Javascript photos all'indice selezionato.

```
function changePhoto(index){
document.images.photoslider.src=photos[index];
}
```

In questo modo, selezionando l'icona numero x, la foto il cui indirizzo è contenuto nella posizione x dell'array photo viene assegnata

all'imagine'photoslider'. L'esempio, completo di tutti i sorgenti, è disponibile sotto la directory FlickrExample1.

Secondo esempio: Rappresentazione ad albero

Il secondo esempio proposto è un po' più complesso del precedente. Sfrutta ancora le API di Flickr, sia pur utilizzando metodi diversi rispetto a quanto visto in precedenza, per accedere ad una serie di foto e visualizzarle. La particolarità sta proprio nella componente che si occupa della visualizzazione. Verrà infatti utilizzato un pacchetto Javascript Ajax per la visualizzazione di grafi. Dato che Flickr fornisce la possibilità di classificare le foto inserite per argomento, l'esempio che verrà illustrato in questo capitolo si occuperà di estrarre le foto di un utente classificate per argomento e di fornirne una rappresentazione ad albero secondo lo schema visualizzato nella figura 7.4.

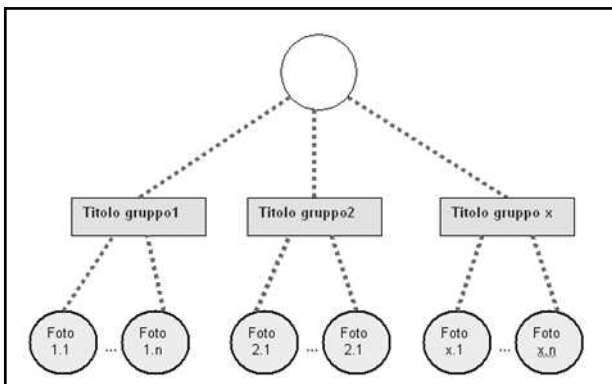



Figura 7.4: Schema di visualizzazione ad albero delle foto.

Quindi, a partire da una radice, che conterrà il nome dell'utente le cui foto sono state richieste, si costruiranno i nodi figli, uno per ogni argomen-

to, che avranno visualizzato il nome dell'argomento. Per ogni nodo "argomento" verranno costruiti i figli che consisteranno in icone rappresentanti le foto corrispondenti e contenenti i link alle foto su Flickr. L'albero così costruito viene visualizzato sullo schermo ed è possibile spostarlo o spostare i vari nodi, ottenendo un effetto molto particolare. E' importante sottolineare come tale effetto dinamico sia ottenuto esclusivamente tramite Javascript, senza utilizzare flash od altri strumenti di animazione. Per una corretta visualizzazione si raccomanda l'utilizzo del borwser Mozilla Firefox.

Introduzione a JSVIZ

JSVIZ è una libreria Ajax (acronimo di Asynchronous Javascript And Xml) open source che permette la rappresentazione di dati sotto forma di grafi. Tali grafi possono essere creati in ogni elemento HTML, anche se, normalmente, vengono posti dentro il tag <body>. La seguente istruzione, per esempio, instanzia un grafo di tipo snowFlake (fiocco di neve)

```
var layout = new SnowflakeLayout( document.body, true );
```

Tramite poi l'oggetto layout si definisce la gestione dell'inserimento di differenti tipi di nodo nel grafo. Per ogni tipo di nodo occorre segnalare al layout come creare un modello ed una vista ("model" and "view") del nuovo nodo. L'attributo model definisce, per il caso snowFlake, le seguenti caratteristiche del nodo:

- childRadius: l'ampiezza del lato dei nodi figli
- fanAngle: l'angolo massimo in cui porre i nodo figli
- rootAngle: l'angolo iniziale del grafo (per i figli viene calcolato automaticamente)

Questi parametri determinano come il nuovo nodo andrà ad interagire con gli altri nodi del grafo. L'attributo "model" ritornerà un oggetto Javascript contenente queste variabili, come nel frammento di codice sottostante.

```
layout.config._default = {
  model: function( dataNode ) {
    return {
      childRadius: 40,
      fanAngle: dataNode.root ? 360: 100,
      rootAngle: 0
    }
  },
```

L'attributo view definisce invece l'aspetto del nodo. Ritorna un elemento DOM. JSViz supporta sia elementi HTML che elementi di tipo SVG. Questa breve disamina non esaurisce certo le caratteristiche del prodotto, ma vuole solo essere una piccola introduzione alla logica ed alla struttura di JSVIZ. Una dettagliata discussione delle caratteristiche di JSVIZ comporterebbe probabilmente un libro intero, data la complessità di questo pacchetto.
Non occorre però perdersi d'animo di fronte a queste considerazioni. Difatti JSVIZ mette a disposizione una classe, XMLTreeLoader, inclusa nella distribuzione, che costruisce automaticamente un grafo a partire dalla struttura gerarchica contenuta in un documento XML.

```
var loader = new XMLTreeLoader( layout.dataGraph );
loader.load( "treedata1.xml" );
```

Per cui con pochi accorgimenti saremo in gradi di utilizzare JSViz senza dover necessariamente approfondire la complessità delle sue funzioni. Nell'esempio costruiremo un file XML ad hoc a partire dalla risposta di flicke e tramite esso costruiremo il grafo di visualizzazione delle foto. Per utilizzare jsviz occorre collegarsi al sito http://jsviz.org e scaricare la libreria per poi scompattarla in una directory posta sotto la root dell'applicazione.

Costruzione della struttura XML

Andiamo ad esaminare il codice che va a costruire il documento

XML per la visualizzazione del grafo delle foto.

```
<?php
require_once("phpFlickr/phpFlickr-2.1.0/phpFlickr.php");

$fh = fopen("FlickrApiKey.txt", "r");
if (!$fh)
{
  die("Non si può trovare la chiave");
}
$api_key = fread($fh, filesize("FlickrApiKey.txt"));
fclose($fh);

$xml_file = '<root>';
$f = new phpFlickr($api_key);

if (!empty($_POST['username']))
{
  $person = $f->people_findByUsername($_POST['username']);
  $groups = $f->people_getPublicGroups($person['id']);
```

Se il parametro 'username', trasmesso dalla form richiedente, non è vuoto, allora, come nell'esempio precedente viene utilizzato il metodo people_findByUsername (che va ad effettuare una chiamata, come detto, al metodo flickr.people.findByUsername) per ottenere l'identificativo che Flickr ha assegnato a tale utente. Ottenuto tale valore si procede con l'invocazione del metodo getPublicGroups per ottenere la lista di gruppi tematici eventualmente utilizzati dall'utente per raggruppare le sue foto. PhpFlickr ci facilita la vita anche stavolta rendendo immediatamente disponibile la lista in un array.

```
  foreach ((array)$groups as $group) {
  $xml_file.='<node txt="'.$group['name'].'" url="no">';
```

```
$photos=$f->groups_pools_getPhotos($group['nsid'],NULL,NULL,NULL,3);
 foreach ((array)$photos['photo'] as $photo) {
  $xml_file.='<node txt="'.$f->buildPhotoURL($photo, "Square");
  $xml_file .= '" url="http://www.flickr.com/photos/';
  $xml_file .= $photo['owner'] . '/' . $photo['id'].'">';
  $xml_file.='</node>';
 }
 $xml_file.='</node>';
 }
}
$xml_file .= '</root>';
```

L'array dei gruppi viene scorso e per ogni gruppo viene creato un nodo con attributi text =”<nome_del_gruppo>” e url=”no”. Per ogni gruppo viene anche effettuata la richiesta delle foto relative, tramite l'invocazione del metodo group_pools_getPhotos e, per ogni foto, viene costruito un nodo con attributi txt=<foto_iconizzata> e text=<url_della_foto>. Il documento XML così costruito viene quindi salvato nel file denominato treedata.xml

```
$myFile = "treedata.xml";
$fh = fopen($myFile, 'w') or die("can't open file");
fwrite($fh, $xml_file);
fclose($fh);
?>
```

L'esempio sottostante riporta il file XML costruito dalla procedura a partire dalla richiesta dei gruppi dell'utente sarahmorton.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
 <node txt="Ghost Towns" url="no">
<node txt="http://farm1.static.flickr.com/206/445779025_2d72a0046d_s.j
```

```
   pg" url="http://www.flickr.com/photos/28066877@N00/445779025"/>
<node txt="http://farm2.static.flickr.com/1247/776654981_cc42f9092f_s.j
   pg" url="http://www.flickr.com/photos/86821176@N00/776654981"/>
<node txt="http://farm2.static.flickr.com/1149/776654983_370ef476e1_s.
   jpg" url="http://www.flickr.com/photos/86821176@N00/776654983"/>
 </node>
 <node txt="Mid-Century, Modern Interiors" url="no">
<node   txt=http://farm2.static.flickr.com/1052/792601182_bc74a98ff7_s.j
    pg url="http://www.flickr.com/photos/66914691@N00/792601182"/>
<node txt="http://farm2.static.flickr.com/1232/791716321_ff419110e0_s.
   jpg" url="http://www.flickr.com/photos/66914691@N00/791716321"/>
<node txt="http://farm2.static.flickr.com/1311/792591644_619e01674c_s
  .jpg" url="http://www.flickr.com/photos/66914691@N00/792591644"/>
 </node>
 <node txt="Wash Your Hands Say Yeah" url="no">
<node txt="http://farm1.static.flickr.com/225/550339722_0e144fcd97_
 s.jpg" url="http://www.flickr.com/photos/23806189@N00/550339722"/>
<node txt="http://farm1.static.flickr.com/62/161384021_ec0e1f1677_
 s.jpg" url="http://www.flickr.com/photos/19873755@N00/161384021"/>
<node txt="http://farm1.static.flickr.com/52/130976947_e40edf8513_
 s.jpg" url="http://www.flickr.com/photos/53525732@N00/130976947"/>
 </node>
</root>
```

Adesso non resta che passare ad esaminare il codice della pagina che, utilizzando Jsviz, costruisce il grafo utilizzando la struttura XML costruita in precedenza. Innanzitutto occorre importare i moduli Javascript di JSVIZ necessari; nell'esempio si ipotizza che tali moduli siano presenti nella sottodirectory jsviz.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3
                                              .org/TR/html4/strict.dtd">
<html>
```

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /
                                                                      >
  <title>JSViz e Flickr</title>

<!-- JSViz Libraries -->
<script language="JavaScript" src="jsviz/physics/ParticleModel.js"/>
<script language="JavaScript" src="jsviz/physics/Magnet.js"/>
<script language="JavaScript" src="jsviz/physics/Spring.js"/>
<script language="JavaScript" src="jsviz/physics/Particle.js"/>
<script language="JavaScript" src="jsviz/physics/RungeKuttaIntegrator.js
                                                                     "/>
<script language="JavaScript" src="jsviz/layout/graph/ForceDirected
                                                            Layout.js"/>
<script language="JavaScript" src="jsviz/layout/view/HTMLGraphView.js"
                                                                      />
<script language="JavaScript" src="jsviz/layout/view/SVGGraphView.js"/
                                                                      >
<script language="JavaScript" src="jsviz/util/Timer.js"/>
<script language="JavaScript" src="jsviz/util/EventHandler.js"/>
<script language="JavaScript" src="jsviz/io/DataGraph.js"/>
<script language="JavaScript" src="jsviz/io/HTTP.js"/>
<script language="JavaScript" src="jsviz/io/XMLTreeLoader.js"/>
```

La parte più significativa del codice è contenuta nella funzione init(), ed è la ridefinizione della “view”, ovvero della modalità di visualizzazione del nodo. Viene, innanzi tutto, effettuata una discriminazione sulla natura del nodo. Se l’attributo url è uguale a «no», allora siamo di fronte ad un nodo che contiene la descrizione del gruppo dentro l’attributo txt. Per cui viene associato al nodo del codice HTML che riporta il contenuto dell’attributo txt ingrassetto e con colore blu.Nel caso in cui, invece, l’attributo url non dovesse essere uguale a « no », il nodo da trattare è un no-

do «figlio», contenente i dati di una foto. Per cui gli viene associato del codice HTML contenente l'immagine iconizzata ed il link alla immagine su Flickr. Le righe in grassetto, nel frammmento di codice seguente, riportano quanto descritto.

```
<script language="JavaScript">
 function init() {
 ...
 {
   var nodeElement = document.createElement( 'div' );
   nodeElement.style.position = "absolute";
   nodeElement.style.width = "22px";
   nodeElement.style.height = "22px";

   if (dataNode.url !="no"){
     nodeElement.innerHTML = '<A HREF="' + dataNode.url + '"><IMG SR
                    C="' + dataNode.txt + '" WIDTH="22"></A>';
   }
   else{
    nodeElement.innerHTML = '<FONT COLOR="BLUE"><B>' + dataNode
                                    .txt + '</B></FONT>';
   }
   nodeElement.onmousedown =  new EventHandler( layout, layout.
                       handleMouseDownEvent, modelNode.id )
   return nodeElement;
 }
}
    }
```

La funzione magnet ritorna un oggetto Javascript con l'attributo minimumDistance settato a 20 pixel. Si tratta della distanza minima che potrà intercorrere tra un nodo e l'altro.

```
...
```

```
layout.forces.magnet = function() {
  return {
    ...
    minimumDistance: 20
  }
...
```

Adesso occorre utilizzare la classe XMLTreeLoader per caricare i dati contenuti nel file treedata.xml nel grafo così definito.

```
var loader = new XMLTreeLoader( layout.dataGraph );
loader.load( "treedata.xml" );
```

Nella figura sottostante (Figura 7.5) ecco come appare l'applicazione utilizzando i dati dell'esempio. Come si può notare l'aspetto ed il comportamento sono molto particolari. Abbiamo utilizzato i servizi di Flickr per costruire un mashup particolarmente originale nel layout. Le classi fornite, contenute nella directory flickerexample2.

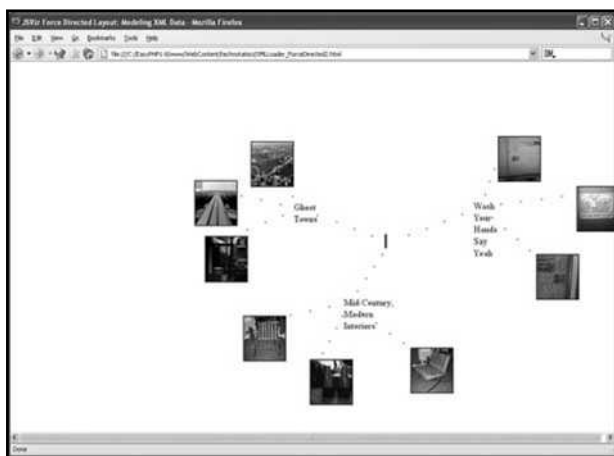



Figura 7.5: Come si presenta l'applicazione.

UN ESEMPIO CON TECHNORATI

Ora faremo la conoscenza di Technorati, il motore di ricerca dedicato ai blogs, e vedremo come utilizzare le interfacce che questo prodotto mette a disposizione per creare una piccola applicazione che ne riprenda i concetti di base.

Introduzione a Technorati

Che cos'è, esattamente, Technorati? Partiamo dalla definizione che ne dà Wikipedia: "*Technorati è un motore di ricerca dedicato al mondo dei blog. Dal dicembre 2005 Technorati indicizza più di 20 milioni di blog. Technorati è stato fondato da Dave Sifry e la sede è presso San Francisco, California, USA.*". Il termine technorati è una parola macedonia, cioè un termine nato dall'unione di due parole: Technological literati (traducibile in italiano come intellettuali tecnologici). L'enorme diffusione che i blog hanno avuto negli ultimi anni ha portato all'esigenza di un motore di ricerca dedicato a questo particolare tipo di siti. Un motore di ricerca che aiutasse a monitorare le conversazioni, a capire chi dice cosa di chi, ad indicare gli argomenti più popolari. Technorati si occupa proprio di questo. Dal punto di vista tecnico – ed in estrema sintesi – potremmo dire che si occupa di individuare ed archiviare i link, permettendo agli utenti di visualizzare quante e quali pagine linkano un determinato indirizzo. Il numero di link che puntano ad una data pagina ne determina il rank, vale a dire l'importanza, il livello di influenza. Technorati misura due differenti tipi di links, che vanno a determinare due tipi di popolarità diverse: gli inbounds links e gli inbounds blogs.
L'esempio di questo capitolo si occuperà proprio di interrogare i servizi Technorati per ottenere, a partire dall'url di un blog fornita come parametro, le informazioni sui blogs che lo referenziano. Informazioni che ci serviranno per creare un grafico dei vari gradi di popolarità di questi blogs. Prima di andare avanti con la presentazione dell'esempio occorre, a questo punto, approfondire brevemente i concetti di inbound links e inbound blogs.

Inbound Links e Inbound Blog: che cosa sono e a che cosa servono

Con il termine inbound links di un sito si intendono i links esterni che puntano al sito stesso. La loro utilità è chiara: servono per fare arrivare navigatori provenienti da altre pagine sul sito, andando così ad incrementarne la popolarità. Più un sito viene visitato più questo diventerà popolare. Il linking esterno ha poi una notevole importanza in quanto fondamentale per poter acquisire un buon piazzamento all'interno dei motori di ricerca.

L'utilizzo che Technorati fa degli inbound links è quello di parametro per misurare la link popularity, ovvero l'indice di popolarità di un sito.

Gli inbound blogs sono concettualmente simili agli inbounds link, ma si riferiscono in maniera specifica ai blog. Gli inbound blogs di un blog (perdonate il gioco di parole) sono quindi i links ad un blog contenuti in altri blog. Vale per gli inbound blogs il discorso fatto per gli inbound links: tanto più un blog è referenziato da altri, tanto piu questo è influente nella blogosfera. Tramite questo parametro Technorati definisce quali sono i blog più influenti nei paesi occidentali. Certo, ci si potrebbe domandare se il fatto che un blog abbia un numero elevato di "inbound blogs" basti automaticamente ad indicarlo come "autorevole ed influente". Probabilmente, per un blog, andrebbe analizzata, oltre alla quantità, anche la qualità dei blog (o dei siti generici) che lo referenziano. Se il mio blog fosse linkato, per dire, dal "Corriere della Sera", questo collegamento dovrebbe ragionevolmente essere pesato diversamente rispetto ad eventuali altri provenienti dai blogs dei miei vicini di casa. Non me ne vogliano questi ultimi.

Al di là di queste doverose considerazioni, occorre prendere atto che, in ogni caso, Technorati è de facto lo standard per la misura della popolarità dei blog.

Dopo questa disamina sui concetti che andremo a trattare non resta che cominciare a lavorare con le API che ci mette a disposizione.

Un percorso obbligato: procurarsi una Key

Come in quasi tutti gli esempi visti sino ad ora, per usufruire dei servizi web che le grandi aziende espongono, occorre procurarsi una chiave. Technorati non differisce, in questo, da altre importanti realtà del web come Google, YouTube o Flickr. Analogamente a Flickr - e a differenza di quanto accade, per esempio, con Google Maps - la chiave fornita non identifica un sito che la utilizza ma un utente. Per cui, una volta acquisita, potrete usarla anche su differenti domini. Si tratta, anche in questo caso, di un codice alfanumerico ed il procedimento da seguire per ottenerla è identico a quanto visto in altri esempi. È necessario, per prima cosa, registrarsi sul sito ed accedere alla pagina *http://www.technorati.com/developers/apikey.html*, riempire la form proposta a video e conservare la stringa generata.

Technorati Cosmos, Richeste e Risposte

Per eseguire query sulle caratteristiche dei blog contenuti nella sua base dati, Technorati ha sviluppato le API Cosmos, di cui si può trovare una dettagliata documentazione all'indirizzo *http://technorati.com/developers/api/cosmos.html*. Non entreremo, nel corso di questo paragrafo, nel dettaglio di queste API, in quanto ci avvarremo di uno strumento, analogamente a quanto visto nel corso dell'esempio su Flickr, che ce le renderà completamente invisibili. È, in ogni caso, necessaria una sintetica rappresentazione del formato di dati fornito da Cosmos per agganciarci ai concetti descritti sino ad ora. Cosmos accetta in input una query contenente, tra gli altri parametri, l'URL di un determinato blog. Altri parametri sono la chiave di riconoscimento (obbligatoria), il limite massimo di risultati da estrarre, il formato con qui questi debbono essere forniti eccetera.
Queste API, quindi, utilizzano un'interfaccia REST, da invocare tramite una GET od un POST HTTP. Per esempio, una chiamate potrebbe essere la seguente:

```
http://api.technorati.com/cosmos?key=<api_key>&url=<url_del_blog>
```

E questo il formato della risposta:

```
<?xml version="1.0" encoding="utf-8"?>
<tapi version="1.0">
<document>
<result>
 <url>URL per cui è stata fatta la richiesta</url>
 <weblog>
  <name>Nome del Blog</name>
  ...
  <inboundblogs>Numero di INBOUND BLOGS</inboundblogs>
  <inboundlinks>Numero di INBOUND LINKS</inboundlinks>
  ...
 </weblog>
 ...
</result>
<item>
 <weblog>
  <name>Nome del Blog che linka quello principale</name>
  <url>URL del Blog</url>
  ...
  <inboundblogs>Numero di INBOUND BLOGS</inboundblogs>
  <inboundlinks>Numero di INBOUND LINKS</inboundlinks>
 ...
 </weblog>
 ...
</item>
... // Lista degli altri blog sotto il tag Item
</document>
</tapi>
```

Il tag <result> contiene i dati del blog il cui indirizzo è stato inviato come parametro. A questo segue un'eventuale lista di entità

<item> contenenti i dati dei blogs che hanno al loro interno links al blog in oggetto. In grassetto i tags <inboundblogs> ed <inboundlinks>, contenenti le informazioni che andremo a prelevare ed il cui significato è stato spiegato nel corso dei paragrafi introduttivi.
Da notare che, nel caso in cui l'URL fornito a Cosmos non risulti essere associato ad un blog, l'elemento <weblog>, e tutti i suoi figli, non compariranno all'interno dell'elemento <result>.

```
<tapi version="1.0">
<document>
<result>
  <url>URL per cui è stata fatta la richiesta</url>
  <inboundblogs>Numero di INBOUND BLOGS</inboundblogs>
  <inboundlinks>Numero di INBOUND LINKS</inboundlinks>
</result>
<item>
  <weblog>
    ...
  </weblog>
  ...
</item>
...
</document>
</tapi>
```

Entriamo adesso nel concreto dell'esempio, presentando lo strumento di cui ci avvarremo per sfruttare i servizi Technorati

Services Technorati, un wrapper per Cosmos

Per comunicare con i servizi Technorati utilizzeremo lo stesso approccio adottato nel caso di Flickr: anziché comporre la richiesta ed effettuare direttamente il parsing della risposta XML, lasceremo che

uno strumento si occupi di effettuare per noi questo lavoro “sporco” restituendoci i dati in un formato intuitivo ed immediatamente utilizzabile. Lo strumento di cui sto parlando è un pacchetto free, chiamato Services Technorati, e disponibile per il download all’indirizzo http://pear.php.net/package/Services_Technorati. Seguire le istruzioni di download (peraltro estremamente dettagliate) per scaricare i files indicati sul proprio computer e, una volta effettuato il download, scompattarli sotto una directory della propria applicazione. Le classi risulteranno immediatamente disponibili. Di fatto Services Technorati fornisce un’interfaccia Object Oriented alle API Technorati,, e con pochissime righe di codice è possibile accedere a tutte le informazioni da esse fornite.

Accesso ai servizi Technorati

Di seguito alcuni frammenti del codice che, utilizzando Services Technorati, interroga le API. Prima di tutto occorre instanziare la classe fornendo come parametro la Key.

```
...
$fh = fopen("TechnoratiKey.txt", "r");
if (!$fh)
{
  die("Non si può trovare la chiave");
}
$techno_key = fread($fh, filesize("TechnoratiKey.txt"));
fclose($fh);

$tapi =& Services_Technorati::factory($techno_key);
...
```

Parlando di interfaccia Object Oriented non si esagerava: per ovviarre ad eventuali modifiche alle API sottostanti, che renderebbero il codice obsoleto, queste vengono accedute tramite un metodo factory.

Una volta instanziato l'oggetto occorre passargli le queries da effettuare ed il risultato sarà contenuto nell'oggetto $technorati_response. Questo altro non è che un array multidimensionale contenente tutte le informazioni ritornate da Cosmos. Per intendersi, Services Technorati parsa il messaggio XML di risposta ed inserisce i valori in un oggetto gerarchizzato di questo tipo:

```
Array(
  [version] => 1.0
  [document] => Array(
      [result] => Array(
           [username] => pippo
            ...
       )
       [item] => Array(
           [0] => Array(
               [weblog] => Array(
                   [name] => Nome Blog 0 ...
                   ...
                   [inboundblogs] => 52
                   [inboundlinks] => 82
                    ...
               )
           )
            [1] => Array(
                [weblog] => Array(
                           ...
                )
            )
            [2] =>...
       )
   )
)
```

Ottenuto questo oggetto passiamo a fornire una rappresentazione grafica dei risultati ottenuti. Rappresenteremo, all'interno di una pagina web, un grafico a barre in cui, per ogni blog ritornato da Cosmos in risposta ad una query, saranno rappresentati gli inbound links e gli inbound blogs.Lo schema di quanto vorremmo ottenere è riportato nella Figura 7.6.

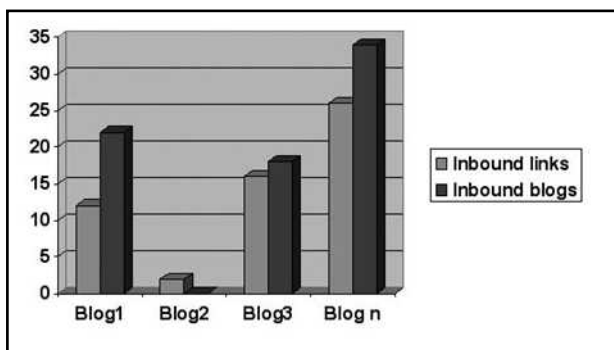



Figura 7.6: Come dovrebbe apparire il grafico dei blogs

Per la costruzione del grafico delle caratteristiche dei blogs verrà utilizzato lostrumento PHP/SWF Charts, disponibile all'indirizzo http://maani.us/xml_chart.

Rappresentazione grafica dei risultati con PHP/SWF Charts

PHP/SWF Charts è un pacchetto estremamente potente che permette la creazione di grafici su pagine web tramite l'utilizzo di PHP e di Macromedia Flash. La distribuzione base è free; ne esiste anche una a pagamento (abbastanza economica, a dire il vero) che fornisce allo sviluppatore alcune features accessorie. In ogni caso, per i nostri scopi, la versione free è più che sufficiente.
Una volta connessi al sito http://maani.us/xml_chart e scaricato il pacchetto, occorre scompattarlo in una directory sottostante la ra-

dice della nostra applicazione.
Per utilizzarlo è sufficiente scrivere due semplici pagine PHP. La prima è la seguente:

```
<HTML>
<BODY>
<?php
include "charts/charts.php";

echo InsertChart ( "charts/charts.swf", "charts/charts_library", "data.php"
, 800, 400 );
?>
</BODY>
</HTML>
```

Di una semplicità estrema. Viene inclusa la classe charts.php, compresa nel pacchetto scaricato. Dopodiché si esegue la direttiva InsertCharts che, come parametri, ha: i cammini ai files charts.swf ed alla directory charts_library (anche questi files sono compresi nella distribuzione); la pagina data.php che contiene i dati da visualizzare e che, tra un attimo, vedremo come costruire; la larghezza e l'altezza in pixel del grafico.
Passiamo alla pagina data.php. Questa si occupa di ricevere l'oggetto ritornato da technorati, di costruire con esso un altro oggetto ad hoc che verrà renderizzato a video dal file charts.swf.

```
<?php
...
$tapi =& Services_Technorati::factory($api_key);
$techno_result = $tapi->cosmos($_POST['url'], NULL);
```

L'oggetto $techno_result viene scorso e si utilizzano i nomi dei blogs e le informazioni contenute nelle sezioni inboundlinks e inbound-

blogs per costruire un altro array multidimensionale, assegnato alla variabile $blogs_chart, così strutturato:

```
$blogs_chart [ 'chart_data' ] = array (
  array ( "", "Nome blog1",     "Nome blog2",  ... , "Nome blog n" ),
  array ( "Inbounds Links",  6,     12,     20,     4  ),
  array ( "Inbounds Blogs",  18,    20,     65,     55  )
);
```

L'ultimo passo consiste nell'inviare l'array così costruito al file charts.swf che ne effettuerà il rendering a video.

```
SendChartData ( $blogs_chart );
```

In figura 7.7 un esempio del risultato che si otterrà a partire dall'inserimento, in una apposita form, dell'URL di un blog.

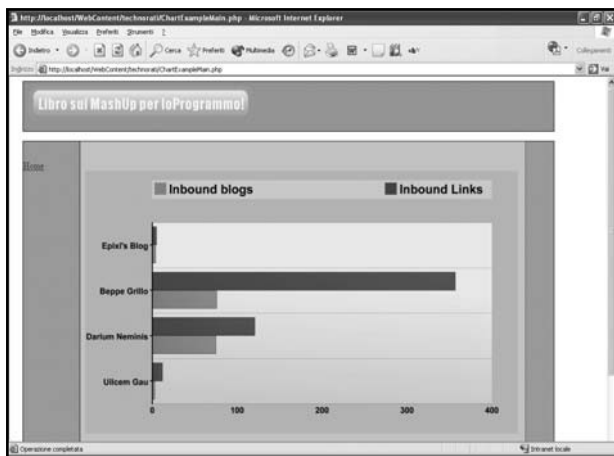



Figura 7.7: Esempio di risultato della nostra applicazione.

ALTRI MASHUP IN JAVA

Anche numerosi siti di e-commerce offrono l'accesso alle proprie funzionalità pubblicando apposite API. Un esempio è E-Bay, ma anche Amazon offre servizi di questo tipo. Di seguito si creerà un mashup tra le ricerche sui siti Web usando le funzionalità di Google e le ricerche sui prodotti (in particolare sui libri) usando i servizi esposti da Amazon. Poi si vedrà come usare i Feed pubblicati da YouTube per creare un mashup di sicuro effetto!

RICERCHE SU AMAZON E GOOGLE

AGoogle ha da sempre la sua punta di diamante nel suo motore di ricerca. Tale motore espone le proprie funzionalità sia come API invocabili lato server che, e questa è una recente novità, permette di usarle anche con delle librerie JavaScript che, grazie ad AJAX, sono utilizzabili lato client senza passare dal server.
Amazon, d'altro canto, offre l'accesso ai suoi tanti prodotti usando opportune API invocabili via SOAP, grazie ad una libreria di sviluppo per Web Services. Nell'esempio che segue si combineranno queste due tecnologie per una ricerca "trasversale" che fa ricerche su siti Web con Google e sui libri (e solo su di essi) venduti da Amazon. Prima di scendere nei dettagli dell'esempio, una breve panoramica delle funzionalità esposte dai due servizi.

E-Commerce Service di Amazon

Amazon offre numerosi Web Services (si veda, per una panoramica, la pagina http://aws.amazon.com). Particolarmente interessante è il servizio E-Commerce Service (d'ora in poi ECS). Grazie ad esso è possibile reperire informazioni sui tanti prodotti venduti da Amazon. Oltre alle informazioni è possibile sfruttare tutte le caratteristiche di un moderno servizio di e-commerce (carrello della spesa, gestione dei vari vendi-

tori, disponibilità in magazzino, prezzi e storico dei prezzi e così via) anche se questa parte, per questo esempio specifico, non verrà sfruttata. In questo momento è importante osservare che tutte le operazioni di ECS non modificano lo stato degli oggetti venduti, ma ne permettono solo la loro gestione (intesa come reperimento, filtro e storico).

Un po' di terminologia

Amazon è evoluto nel tempo fino a diventare un aggregatore di servizi e uno spazio di vendita di prodotti non solo suoi ma di un'ampia gamma di fornitori, alcuni dei quali con delle caratteristiche "privilegiate" rispetto ad altri dovute a loro posizioni di mercato dominanti o a servizi esclusivi, ma che, almeno in principio, non esclude che chiunque possa usare Amazon come luogo di vendita dei propri prodotti e/o servizi. In questo senso Amazon indica con il termine "marketplace" la totalità dei prodotti e/o servizi vendibili dal suo portale. I servizi ECS permettono l'accesso solo ad un sottoinsieme di prodotti che sono quelli relativi ai venditori precedentemente indicati come "privilegiati" e che vengono chiamati "Pro Merchant Sellers" e "Merchant@ vendors". Come fornitori di un servizio di accesso ai prodotti venduti da Amazon possiamo avere delle commissioni sul venduto (in questo caso diventiamo degli "associati").

Cercare gli oggetti

Ciascun oggetto venduto da Amazon ha un proprio identificativo univoco; esso è chiamato ASIN (iniziali di "Amazon Standard Item Number") e per i libri, caso che ci interessa per l'esempio, esso coincide con il codice ISBN. L'operazione di ricerca principale in ECS è chiamata ItemSearch; essa permette di impostare le caratteristiche desiderate e di reperire tutti gli oggetti che soddisfano a tali caratteristiche. Ma, in concreto, come si realizza una simile interrogazione?

Ottenere la chiave di accesso

Come gran parte dei servizi offerti da Amazon, anche ECS può

essere interrogato dopo essersi registrati al servizio. La registrazione, tra le altre cose, permette di recuperare il valore di una chiave. Tale chiave va associata ad ogni invocazione fatta dall'applicazione che si realizza verso il server di Amazon.
ECS, a differenza di altri servizi, è completamente gratuito. Però è necessario prestare particolare attenzione alle sue clausole di utilizzo (si veda
http://www.amazon.com/AWS-License-home-page-Money/b/ref=sc_fe_c_0_12738641_5/103-2085190-0787832?ie=UTF8&node=3440661&no=12738641 per i dettagli). Tra le altre cose è vietato memorizzare in maniera permanente le informazioni accedute usando il servizio (è permessa una copia locale per un periodo di tempo che varia dal tipo di dato: per alcuni dati si può solo fare una copia cache per non più di 1 mese, per altri non più di 24 ore!) e sono molto rigide le regole che specificano come usare le informazioni sui prezzi che si mostreranno all'utente. Per registrarsi, collegarsi alla pagina *http://aws.amazon.com/* e seguire il link "Click here to sign up". La registrazione prevede l'inserimento della propria email e, se si è utenti di Amazon, della propria password, altrimenti verrà creato un nuovo account. Terminata la registrazione verrà inviata una mail con i dettagli per ottenere la chiave e attivarla. Una volta attivato l'account siamo pronti perla generazione dei client per l'accesso ai servizi.

Generare gli stub del Web Service

Quando si vuol realizzare un Web Service con Java, bisogna scegliere il framework o la tecnologia di accesso. Per l'esempio si è scelto di usare XFire, ma i passi sono molti simili nel caso si scelga di utilizzare un altro framework come, per esempio, Axis. Di fondamentale importanza è reperire il WSDL che descrive tutti i dettagli del servizio. Amazon mette a disposizione il WSDL di ECS alla pagina http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl. I passi per la generazione degli stub con XFire (passi che vanno inseriti nella console di comando o in un file .BAT) consistono nell'inizializzare opportunamente il classpath con tutte le librerie:

```
SET XFIRE_HOME=.
SET LIB=%XFIRE_HOME%/lib
SET GPATH=%XFIRE_HOME%/xfire-all-1.2.6.jar;%LIB%/jaxb-api-2.0.jar;\
 %LIB%/stax-api-1.0.1.jar;%LIB%/jdom-1.0.jar;%LIB%/jaxb-impl-2.0.1.jar;
                                                                     \
 %LIB%/jaxb-xjc-2.0.1.jar;%LIB%/wstx-asl-3.2.0.jar;\
 %LIB%/commons-logging-1.0.4.jar;%LIB%/activation-1.1.jar;\
 %LIB%/wsdl4j-1.6.1.jar;%LIB%/XmlSchema-1.1.jar;\
 %LIB%/xfire-jsr181-api-1.0-M1.jar;%ANT_HOME%/lib/ant.jar
```

Per poi eseguire WsGen con gli opportuni parametri (far riferimento alla documentazione di XFire per i dettagli):

```
java -cp %GPATH%; org.codehaus.xfire.gen.WsGen -wsdl \
http://webservices.amazon.com/AWSECommerceService/AWSECommerce
                                              Service.wsdl? \
 -o . -p it.ioprogrammo.mashup.wsclient.amazon -overwrite true
```

Si noti che lo script specifica anche come dovrà essere formato il package delle classi generate.

Attenzione
Con il termine "stub" si intende la costruzione dell'infrastruttura per la comunicazione con il servizio remoto. In particolare gli stub client sono pronti per essere usati per inviare i messaggi e leggere il contenuto della risposta usando le strutture dati del linguaggio (in questo caso Java) senza preoccuparsi dei dettagli (né di comunicazione, in questo caso invocazioni http/https, né di rappresentazione dei messaggi, in questo caso messaggi SOAP).

Il risultato del comando WsGen è la generazione di tutte le classi che permettono di accedere al servizio remoto di ECS (Figura 8.1).

Si noti che tra il nome del package figura anche la data che permette di identificare la versione del servizio usato (in questo caso 2007-06-13, ovvero il 13 giugno del 2007, essendo il formato della data espresso secondo la notazione statunitense).

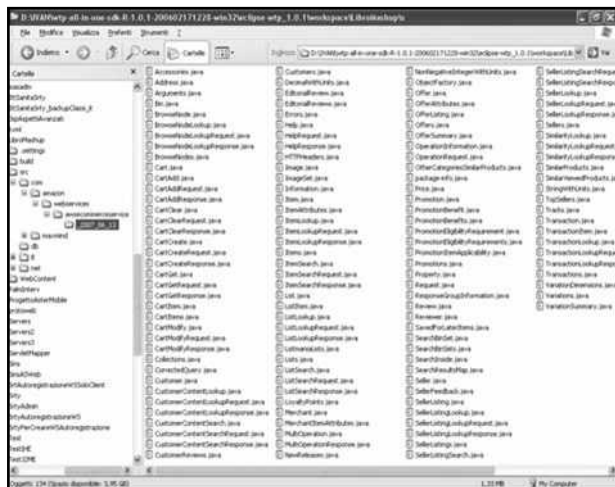



Figura 8.1: Tutte le classi generate da XFire per l'accesso a ECS.

La parte server dell'esempio

La classe TestAmazon (del package it.ioprogrammo.mashup.wsclient) permette di invocare ECS e ricercare una parola chiave tra i libri grazie al metodo; ecco i dettagli di tale metodo. Il servizio di Amazon che viene usato è ItemSearch. Esso viene invocato con un opportuno parametro, di tipo OtemSearchRequest, che descrive i filtri desiderati (ovvero le keyword passate al metodo cercaLibri, e l'indice dei libri):

```
public static java.util.List<Item> cercaLibri(String keyword){
 AWSECommerceServiceClient client = new AWSECommerceServiceClient()
;
```

```
AWSECommerceServicePortType port =
   client.getAWSECommerceServicePort();
 ItemSearch isearch = new ItemSearch();
 isearch.setAWSAccessKeyId(keyId);
 ItemSearchRequest irequest = new ItemSearchRequest();
 irequest.setSearchIndex("Books");
 irequest.setKeywords( keyword );
 irequest.setSort("daterank");
 isearch.setShared( irequest );
 ItemSearchResponse iresponse = port.itemSearch(isearch);
 java.util.List<Items> libri = iresponse.getItems();
 if (libri.size()>0)
   return libri.get(0).getItem();
 else
   return new java.util.ArrayList();
}
```

È stato evidenziato il codice che imposta i filtri per la ricerca: nel caso specifico viene usato l'archivio dei libri, ricercate le parole chiave passate come argomento del metodo e si richiede un ordinamento dei dati per data di pubblicazione (altri tipi di ordinamento sono reperibili nella documentazione ufficiale e, in particolare, consultando la pagina http://docs.amazonwebservices.com/AWSECommerceService/2007-06-13/DG/USSortValuesArticle.html#USSortValuesArticle_books).

Il client e le ricerche

La parte JSP deve prevedere una pagina che si occuperà di chiedere all'utente le parole da ricercare e di innescare le ricerche sulle diverse fonti. Il primo passo è quello di includere i riferimenti alle librerie JavaScript di Google AJAX Api (per i dettagli del servizio si veda http://code.google.com/apis/ajaxsearch/)

```
<script src=
```

```
"http://www.google.com/uds/api?file=uds.js&amp;v=1.0&amp;key=
                                                      chiave"
type="text/javascript">
</script>
```

Inoltre è necessario inizializzare e configurare opportunamente il servizio di ricerca; infatti tale servizio può ricercare sulle pagine Web, sui blog, sui video o aziende localizzate su Google Maps. Ecco come impostare la ricerca sulle sole pagine Web usando, per l'esempio che si realizza, una funzione chiamata OnLoad:

```
<script language="Javascript" type="text/javascript">
//<![CDATA[
 function OnLoad() {
   var searchControl = new GSearchControl();
   var options = new GsearcherOptions();
   options.setExpandMode(GSearchControl.EXPAND_MODE_OPEN);

   searchControl.addSearcher(new GwebSearch(), options);
   searchControl.setResultSetSize(GSearch.LARGE_RESULTSET);
   searchControl.draw(document.getElementById("searchcontrol"));
   searchControl.setSearchStartingCallback(this, amazonSearch);
 }
 GSearch.setOnLoadCallback(OnLoad);
//]]>
</script>
```

Analizzando la riga evidenziata, si nota che è stato settato un "callback" all'inizio di ogni nuova ricerca. In pratica è stata dichiarata una funzione "amazonSearch" che va invocata all'inizio di ogni nuova ricerca. Questo perché si vuol far eseguire l'invocazione dei Web Services di Amazon ogni volta che l'utente ricerca una parola con Google; la funzione è la seguente:

```
function amazonSearch(searchControl, searcher, query) {
  var invoca = "amazon.jsp?query="+query;
  GDownloadUrl(invoca, function(doc) {
    document.getElementById("searchamazon").innerHTML = doc;
     });
  return true;
}
```

Attenzione
Alla pagina http://code.google.com/apis/ajaxsearch/wizards.html è presente un Wizard estremamente interessante: esso permette di generare il codice JavaScript adatto ad effettuare alcuni tipi di ricerca, personalizzando anche il layout del risultato. È un ottimo modo per prendere confidenza con le caratteristiche del servizio, usando un approccio "learn by example".

Non resta che analizzare la pagina invocata (usando la solita funzione GDownloadUrl esposta da Google Maps!), ovvero la pagina amazon.jsp; essa non fa altro che passare la query ricevuta come parametro al metodo cercaLibri visto in precedenza:

```
<%@page import="it.ioprogrammo.mashup.wsclient.*"%>
<%@page import="com.amazon.webservices.awsecommerce
                                        service._2007_06_13.*"%>

<%
  String query = request.getParameter("query");
  java.util.List<Item> libri = TestAmazon.cercaLibri(query);

  System.out.println("amazon.jsp: "+libri.size()+"elementi");

  if (libri.size()>0){
```

```
 // stampa i risultati
 }else{
%>
 Nessun risultato...
<%
 }
%>
```

Di seguito il codice che stampa i dettagli dei risultati reperiti (tra cui un ciclo che scorre e stampa la lista degli autori del libro):

```
  for(int ix=0; ix<libri.size() && ix<20; ix++){
  Item it= libri.get(ix);

  ItemAttributes att=it.getItemAttributes();
%>
  <p>Titolo:
  <a href="<%= it.getDetailPageURL() %>"
 target="_blank"><%= att.getTitle() %></a><br />
  Autori:
<%

  java.util.List<String> autori = att.getAuthor();
  for(int i=0; i<autori.size(); i++)
    out.print(autori.get(i));
%>
  <br />
  ISBN: <%= it.getASIN() %><br />
  </p>
<%
  }
```

In Figura 8.2 un esempio, in cui è stata cercata la parola AJAX.

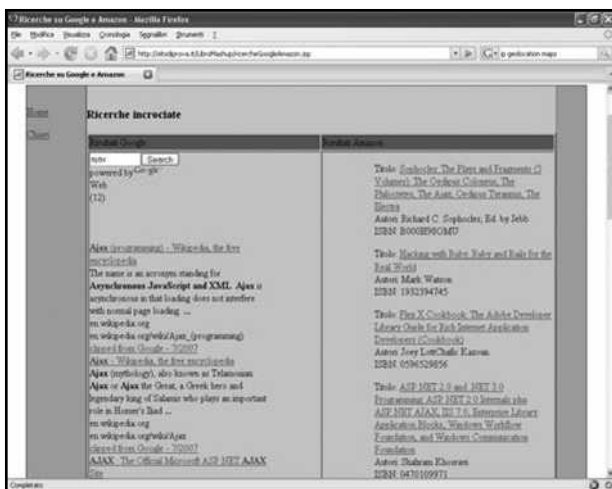



Figura 8.2: Ricerca di AJAX, sia con Goole che sui libri di Amazon.

Conclusioni sui Web Services

Quest'ultimo esempio ha mostrato come due tecnologie, AJAX e i Web Services, possano convivere per creare applicazioni mashup. Il limite attuale per l'uso di AJAX è la possibile incompatibilità tra browser (soprattutto per quelli obsoleti) ma è un limite destinato a scomparire, man mano che evolvono framework che incapsulano la logica di interscambio e gestione della comunicazione. Per i Web Services è necessario appoggiarsi ad un framework esterno, per evitare di perdersi nei tanti dettagli altrimenti necessari per gestire a basso livello la comunicazione tra client e server. Ma per gli sviluppatori entrambe le tecnologie sono un concreto vantaggio, sia per rendere più semplice l'uso dell'applicazione per gli utenti finali (AJAX), sia per demandare a un servizio esterno gran parte della logica o del reperimento di dati necessari al corretto funzionamento del proprio applicativo (Web Services).

IL FUTURO DEI MASHUP…

I mashup, siano essi semplici, complessi, che fanno uso di librerie di terze parti o siano sviluppati con tecnologie proprietarie, stanno invadendo il Web. Si è aperto il libro citando numerosi servizi per non programmatori. L'intero libro è dedicato a chi programma e ha descritto alcuni dei tanti frame work e librerie disponibili. Anche altri "big" dell'informatica si stanno adoperando per fornire framework che, a vario livello, aiutano a creare mashup. IBM, per esempio, ha iniziato "AJAX Toolkit Framework" (*http://www.alphaworks.ibm.com/tech/ajaxtk*) che ben presto è divenuto uno dei progetti di Eclipse (*http://www.eclipse.org/atf/*). La Sun ha recentemente rilasciato una nuova versione del suo "Sun Web Developer Pack" (sito di riferimento *http://developers.sun.com/web/swdp/*, per il download *http://developers.sun.com/web/swdp/docs/tutorial/download.html*, Figura 8.7). In esso trovano posto tante tecnologie emergenti, tra le quali:

- *jMaki*: un framework che, grazie a template e a widget AJAX, permette di costruire applicazioni Web 2.0 in maniera modulare e semplificata (esistono widget già pronti per i principali servizi!) con plug in per integrare il frame work nei principali IDE esistenti;

- *Dynamic Faces*: estende le Java Server Faces per far sì che l'interfaccia supporti il modello AJAX; il tutto senza cambiare gli usuali componenti JSF con cui si sono costruite le applicazioni Web "tradizionali";

- Lightweight Programming Models usando linguaggi dinamici all'interno della piattaforma Java;

- RESTful Web Services, per la realizzazione di servizi Web se-

condo il modello REST;

- *ROME Propono*: per la creazione, organizzazione e gestione di feed Atom.

Figura 8.7: Dalla Sun il suo "Sun Web Developer Pack".

Questo proliferare di librerie e tecnologie fanno sì che i mashup si stiano imponendo sempre più nel panorama informatico mondiale e non c'è dubbio che ogni programmatore di applicazioni Web debba, prima o poi, confrontarsi con tale approccio. Speriamo che questo libro sia un'utile base su cui costruire tanti e nuovi mashup innovativi!